SCOPE: AN INTRODUCTORY GRAPHICS LANGUAGE

Scott Hilmar Mayer

United States
Naval Postgraduate School

THESIS

SCOPE: AN INTRODUCTORY GRAPHICS LANGUAGE

by

Scott Hilmar Mayer

June 1970

SCOPE: An Introductory Graphics Language

by

Scott Hilmar Mayer
Lieutenant, junior grade, United States Navy
B.S., University of Illinois, 1969

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1970

ABSTRACT

The SCOPE language has been designed to provide an introduction to interactive computing and the cathode-ray tube graphics display. The user is given the opportunity to input a figure and see the kinds of things that can be done with that figure on the display screen. The language has been implemented on an XDS 9300 computer interfaced with an Adage AGT 10 graphics terminal. Each instruction is described,and the algorithms used to actually display figures are also described. Suggestions for future implementations are also included.

TABLE OF CONTENTS

I. INTRODUCTION ---------- 7

II. PROPERTIES OF SCOPE ---------- 9

A. FIGURE DEFINITION ---------- 9
B. HIDDEN LINES ---------- 10
C. WIRE FRAME ON SOLID REPRESENTATION ---------- 10
D. ROTATION ---------- 11
E. RELOCATION ---------- 11
F. SIZE VARIATION ---------- 12
G. ADDITION AND SUBTRACTION OF LINES ---------- 12
H. OTHER PROPERTIES ---------- 12

III. BACKGROUND ---------- 14

IV. SUMMARY AND CONCLUSION ---------- 19

APPENDIX A SYNTAX OF SCOPE INSTRUCTIONS ---------- 21

I. DEFINITIONS ---------- 22

II. SCOPE INSTRUCTION ---------- 26

A. STORAGE MANIPULATION INSTRUCTIONS ---------- 26

1. Define Block ---------- 26
2. Erase Instruction ---------- 28
3. Input Instruction ---------- 28
4. Load Instruction ---------- 29
5. Output Instruction ---------- 29
6. Write Instruction ---------- 29

B. DISPLAY FORMAT INSTRUCTIONS ---------- 30

1. Clear Instruction ---------- 30
2. Dash Instruction ---------- 30
3. Intensity Instruction ---------- 31
4. Solid Instruction ---------- 31
5. Undash Instruction ---------- 31

C. FIGURE ORIENTATION INSTRUCTIONS ---------- 31

1. Face Orientation Instructions ---------- 31
2. Expand Instruction ---------- 32
3. Expand to Instruction ---------- 32
4. Move Instruction ---------- 32
5. Move to Instruction ---------- 32
6. Rotate Instruction ---------- 33

7. Rotate to Instruction ----------------- 33
8. Shrink Instruction -------------------- 33
9. Shrink to Instruction ----------------- 34

D. LINE CONTROL INSTRUCTIONS ----------------- 34

1. Add Instruction ----------------------- 34
2. Hide Instruction ---------------------- 34
3. Show Instruction ---------------------- 35
4. Subtract Instruction ------------------ 35

E. MISCELLANEOUS INSTRUCTIONS ---------------- 35

1. Done Instruction ---------------------- 35
2. Get Instruction ----------------------- 36
3. Stop Instruction ---------------------- 36
4. ZAP Instruction ----------------------- 36

APPENDIX B IMPLEMENTATION -------------------------- 37

I. STORAGE ALLOCATION -------------------------- 40

II. ROTATION INSTRUCTION ------------------------ 42

III. FIGURE DISPLAY ------------------------------ 45

COMPUTER LISTING ---------------------------------- 47

LIST OF REFERENCES -------------------------------- 94

INITIAL DISTRIBUTION LIST ------------------------- 95

FORM DD 1473 -------------------------------------- 97

ACKNOWLEDGEMENT

The author wishes to express his appreciation for the criticisms and helpful suggestions offered by Professor G. L. Barksdale.

I. INTRODUCTION

In recent years, one of the most difficult problems for computer scientists to solve has been the input-output problem. Computer memories and processing units have become very fast and reliable as electronic sophistication has increased. The input-output, however, is usually dependent on mechanical equipment such as card readers and punches, paper tape readers and punches, magnetic tape units, and line printers. The maximum speed and efficiency of this mechanical equipment will never compare favorably with the speeds and reliability of the electronic circuits the mechanical equipment service. The necessity for motion in the mechanical equipment that does not exist for the electronic circuits is, in fact, the property that inherently makes the input-output processes very slow compared to the other computer functions. The natural solution to this problem is to develop input-output equipment that is not mechanical, and in fact the cathode-ray tube has proven to be very successful.

Most, if not all, beginning programers are trained to punch cards and get their output in the form of a computer printout from the line printer, and they often do not realize the benefits of interactive computation that are possible with a graphics display. A basic demonstration language is necessary to acquaint users with the properties and possibilities of a graphics display. It should not only show the user what he can do; it must also show him the kinds of input he must provide to make the system useful.

The purpose of this thesis is to provide this kind of language. SCOPE, as it is called, has been implemented in FORTRAN IV on the Xerox Data System 9300 computer interfaced to an Adage AGT 10 graphics terminal at the Naval Postgraduate School.

II. PROPERTIES OF SCOPE

SCOPE has been designed to show a beginning user of the graphics display various three-dimensional picture processing capabilities that will be useful to him in future problems. At the same time it will be possible for him to get an introduction to real-time interaction with the computer if the graphics display is part of a terminal to the computer. SCOPE instructions cannot only originate from cards, but also from the terminal itself, and errors in the instructions can be corrected on line at the terminal. The following paragraphs describe the properties of SCOPE. A list of the specific instructions with acceptable abbreviations is presented in Appendix 1.

A. FIGURE DEFINITION

It is necessary for a programer to understand what kind of information he must provide to the computer so that it can be possible to use the computer to manipulate the figure. SCOPE requires that the user define each side of a figure by giving the three-dimensional rectangular coordinates of each point on the side, and indicate whether there should be a "move" or a "draw" between the points. This is accomplished by saying whether each line will be shown or hidden for various orientations of the figure. For example, it is possible for a line on the front side to be shown when the front side is shown by itself, but not have the line show when the front

side is shown in combination with the top side and the left side. The hidden line problem that is very important in graphics is also introduced at the same time.

B. HIDDEN LINES

The problem of determining which lines of a figure will show in any possible orientation of a figure has proven to be very difficult. For specific figures the problem has solutions, but for the general case there is a definite opportunity for further research. It is very important that a beginning user realize the existence of this problem, and SCOPE offers this opportunity. Besides the properties already mentioned, SCOPE also allows the user to hide or show any line on the figure after he has defined it, and that line will remain hidden or show every time that particular combination of sides occurs.

C. WIRE FRAME OR SOLID REPRESENTATION

Every user has had experience with both clear items and solid items, and the graphics display can be used to represent any figure as either a solid opaque item or a transparent item in which every line can be seen. SCOPE allows the user to observe the figure in either orientation and further allows the option to show the back portion of a clear wire frame picture with either dashed or solid lines.

D. ROTATION

SCOPE allows two kinds of rotations: absolute and relative. The absolute rotation is a rotation that first returns the figure to the original orientation defined by the user, and the rotation is then made relative to that orientation. The relative rotation is applied directly to the figure as it is oriented on the screen when the rotation command is issued. Both of these rotations are made relative to a fixed set of orthogonal axes.

Closely related to the rotation instructions are the side instructions. These instructions allow the user to specify any side of the figure, and the figure on the screen will be the two-dimensional projection of that side as it was defined by the user. The side instructions, along with the rotation instructions, are very valuable to further show the user the kinds of applications for which the graphics display can be used effectively.

E. RELOCATION

Like the rotations, there are two kinds of relocation: absolute and a relative. The absolute relocations are made relative to the center of the screen. The relative relocations, like relative rotations, are applied directly to the figure as it is oriented on the screen when the command is issued. These instructions further demonstrate the versatility of the graphics display.

F. SIZE VARIATION

SCOPE gives the user not only the option to make his figure larger or smaller, but also the option to do it absolutely or relatively. The relative instructions change the size of the figure by changing the scale of the figure as it exists when the instructions are issued. The absolute instructions are used to give a specific value to the scale and can be treated as being relative to a scale value of zero. By being able to change size, the beginning user will see the possibility of focusing on specific parts of a picture, a very important function of a graphics display.

G. ADDITION AND SUBTRACTION OF LINES

SCOPE gives the user the option to add lines to any side of the figure and also allows the same options to hide or show the lines that were available in the original figure definition. SCOPE further allows subtraction of lines from any side. A subtracted line will not appear in any orientation of the figure. This facility allows for correction of errors that can occur when a figure is defined.

H. OTHER PROPERTIES

SCOPE also has instructions that allow the user to change the brightness of a figure and to have the representation of any side consist of dashed or solid lines for all orientations of the figure.

The only limitation to the number of different figures a user can have at one time is the amount of available storage.

F. SIZE VARIATION

SCOPE gives the user not only the option to make his figure larger or smaller, but also the option to do it absolutely or relatively. The relative instructions change the size of the figure by changing the scale of the figure as it exists when the instructions are issued. The absolute instructions are used to give a specific value to the scale and can be treated as being relative to a scale value of zero. By being able to change size, the beginning user will see the possibility of focusing on specific parts of a picture, a very important function of a graphics display.

G. ADDITION AND SUBTRACTION OF LINES

SCOPE gives the user the option to add lines to any side of the figure and also allows the same options to hide or show the lines that were available in the original figure definition. SCOPE further allows subtraction of lines from any side. A subtracted line will not appear in any orientation of the figure. This facility allows for correction of errors that can occur when a figure is defined.

H. OTHER PROPERTIES

SCOPE also has instructions that allow the user to change the brightness of a figure and to have the representation of any side consist of dashed or solid lines for all orientations of the figure.

The only limitation to the number of different figures a user can have at one time is the amount of available storage.

The user can work with only one figure at a time, but he can work with any figure he has defined. He may also erase any figure from storage to make room for new figures, and he has the option to save up-dated figures on peripheral storage such as magnetic tape and paper tape, and reload them at a later time.

Errors may be corrected at the graphics terminal, and the error messages are displayed on the screen to facilitate error correction. These facilities are very necessary for an introductory interactive graphics language.

III. BACKGROUND

Before FORTRAN was developed, computer users had to program in complicated machine languages. FORTRAN was developed as a general purpose engineering language. Similar attempts are presently being considered to develop higher level graphics languages and graphics subroutine packages. A great portion of this work has been done in engineering design systems.

The computer has been shown to be very valuable for the engineering designer. Mechanical drawings can be updated on a graphics screen, and detailed analysis of blown up portions of designs can easily be done. It is possible to eliminate many of the construction difficulties that would normally not be discovered until an expensive model is built. Further, it is also possible to simulate the operation of an item being designed on the display. For example, there has been work to simulate carrier landings of new airplanes to determine the pilot's field of visability [Siders 1966]. The major difficulty with these systems such as General Motor's DAC-I system and North American Aviation's AUTODRAFT system is that they have been designed for specific hardware configurations and are not readily adapted to other systems [Siders 1966].

There are, however, systems that are somewhat more flexible being developed. The comprehensive SKETCHPAD system, designed at the Massachusetts Institute of Technology by

I. E. Sutherland, is considered to be the pioneer effort in this area [Siders 1966]. A more recent system is the POGO system, developed at the Rand Corporation [Boehm 1969]. In fact, "Permitting programmers to create graphics programs without spending a great deal of time learning the intricacies of the graphics-subroutine package" is one of the major goals of the POGO system [Boehm 1969]. Another system from M.I.T. uses a language called GRAPHSYS which "provides a convenient, high-level, and nearly display-independent interface between the user and the Display Controller" [Thornhill 1968]. The tendency in each of these systems is to allow the user to solve different kinds of graphics problems without having to become an expert in the hardware and machine language of the system he is using. If the user changes systems, he will be able to use the new system almost immediately if the higher level language is implemented on the new system.

The graphics subroutine packages provided with existing languages are also very useful. IBM, for example, provides the <u>IBM SYSTEM/360 OPERATING SYSTEM GRAPHICS SUBROUTINE PACKAGE (GSP) FOR FORTRAN IV, COBOL, AND PL/I</u> described in Ref. 7. Similarly, the Rand Corporation has designed <u>THE INTEGRATED GRAPHICS SYSTEM FOR THE IBM 2250</u> which runs on an IBM 360/40. IGS may be used with programs written in FORTRAN, PL/I, SIMSCRIPT 1.5, and assembly language [Ref. 2]. These packages allow the user to define his own algorithms in a

language he is familiar with and still be able to use the capabilities of a graphics system. They are especially valuable when they are system-independent for the same reasons that machine-independent languages are valuable. To achieve standardization, however, there is a need to define the properties a good graphics system should have.

One way to determine a set of standard properties is to see what has been done on various systems. For example, in 1965, a system at Lockheed-Georgia had the following capabilities [Siders 1966].

1. Four views: three principle projections and, optionally, either an isometric or perspective.
2. Conversion to display any desired view and return to four views on request.
3. Definition of points.
4. Definition of lines.
5. Definition of conics.
6. Changing scale.
7. Rotation about designated axis.
8. Translation.
9. Free-hand sketching.
10. Alphanumeric display.
11. Deletion.

There are other properties that have been developed on other systems that are also worth consideration:

1. The ability to select a specific graphic technique from a menu; a list of options.
2. The ability to generate an entire figure from a small segment. This feature is useful when designing symmetrical items like gears.

3. The ability to recall figures previously defined in order to combine them with present figures. This feature is desirable when items like screws and rivets have to be shown.

4. The ability to focus on specific portions of figures and make them larger.

There are, of course, many other useful properties that can be included, but problems exist that will have to be solved to design an effective general system. The problem of inputting the data points for the basic figure into the computer is one of these difficulties. It is a very tedious process to have to define each point in a view, but the computer must ultimately have just this information.

For figures that require only two dimensions it is relatively easy to optically scan engineering drawings or microfilms to input figures. The RAND TABLET, used with the POGO language [Boehm 1969] and the IGS package [Brown 1968], has been designed as a solution for this problem along with various kinds of digitizers. Other systems allow the user to sketch figures on the display with a light pen. The computer can be programmed to take the output from this equipment and put it in a form compatible with the data structure used internally by the computer. Some systems also allow users to input figures by defining functions that will generate the data points.

It is much more difficult, however, to input figures that will eventually be presented in a three-dimensional orientation. This is because the computer has to be programmed to take two-dimensional views and create three-dimensional

coordinates. To determine the third coordinate of a line in one view, that line must be found in another view in a different plane. This is a very difficult, it not impossible, problem to solve for any general routine. SCOPE has been designed to introduce the beginner to the input problem for three-dimensional figures.

IV. SUMMARY AND CONCLUSIONS

SCOPE, which has been designed to introduce the user to the problems of interactive graphics, can be used very successfully by the beginning user in that capacity. SCOPE, however, does not introduce the user to many of the techniques and equipment that are unique to graphics.

For example, the user is not introduced to the light pen. The light pen can be used to add lines to a figure on the screen, subtract lines from a figure, or change the position of a line. The modified figure can then be saved in the computer. SCOPE requires that instructions be submitted to add and subtract lines which can be a very tedious task.

The light pen can also be used for menu selection. A menu is a list of possible graphics activities displayed on the screen. The user normally selects one by pointing the light pen at the item on the list, and the appropriate routines in the computer are then activated to process that activity. A beginning user should certainly be introduced to the use of menus even if he does not get an opportunity to actually try one with SCOPE.

SCOPE also does not introduce the user to the use of the joystick. The joystick is normally used to translate figures, rotate them, or change the intensity by merely moving the joystick or rotating it. It should be possible to save any modifications that have been made on a figure by the joystick.

Like the light pen and joystick, the function switches are very versatile. They can be programmed to automatically signal any graphics function, and they can be modified at any time to assign a different use to each one.

Another graphics technique that SCOPE does not introduce is continuous motion. For example, it is possible to rotate a figure continuously in small increments so the figure appears to be spinning. It is also possible to make a figure appear as if it is moving across the screen, and it is even possible to make a figure appear to be moving toward and away from the user by changing its size and intensity.

SCOPE also does not introduce the concept of combining two or more figures to create a third figure. This skill is used very often in studying trees and other list-processing applications.

Future extensions of SCOPE should certainly contain an introduction to these techniques and any other techniques that might be valuable to the beginning user.

APPENDIX A

SYNTAX OF SCOPE INSTRUCTIONS

SCOPE has been designed to make the instructions free format. If desired, more than one instruction can be submitted at a time. The only limitations are that each instruction must be submitted in its entirety, and each instruction must be followed by a dollar sign. Any place where a blank is required or optional, any number of blanks can be submitted. Blanks are not allowed within numbers, but they are allowed between the number and the sign, and they are allowed between the number and the comma or dollar sign following it. Also, there are abbreviations for each instruction that can minimize typing by the user and save space on the input line.

All SCOPE instructions are made relative to the conventional right-hand, three-dimensional rectangular coordinates which remain fixed, with the origin placed in the center of the screen. The positive end of the X axis points toward the user, the Y axis is horizontal, with the positive end pointing to the user's right, and the Z axis is vertical with the positive end pointing up. The picture on the screen presented to the user will always be the projection of the figure on the Y-Z plane defined by these fixed, reference axes.

I. DEFINITIONS

The following definitions of basic SCOPE elements will be used throughout the definitions of the SCOPE instructions. The symbol "::=" will be defined to mean "is equivalent to," and "b" will be used in all locations where blanks are optional. Required blanks will be shown as blanks in the text.

A. (NUMBER) A NUMBER is a signed integer. If no sign appears, the number is assumed to be positive. Otherwise the sign of the number is determined by the right-most sign. For example, -+6 will be stored as +6 while +-6 will be stored as -6. All NUMBERS must be less than $2^{24}-1$ and greater than -2^{24} on the implementation at the Naval Postgraduate School.

B. (VIEW) A VIEW can be any one of the possible faces of the figure. A VIEW is presented in the following format.

VIEW	ABBREVIATION
FRONT	F
BACK	B
RIGHT	R
LEFT	L
TOP	T
BOTTOM	BO

C. (AXIS) An AXIS can be any one of the axis names X, Y, or Z and refers to the reference axes.

D. (NAME) A NAME consists of at least four characters, and all characters except the comma and dollar sign are allowed. Blanks are allowed in any character position except the left-most. Only the four left-most characters are stored internally by the computer, which means that any two figures that have NAMES beginning with the same four characters will be treated as if they have exactly the same name.

E. (DIRECTION) A DIRECTION is taken from the following list.

DIRECTION	ABBREVIATION
RIGHT	R
LEFT	L
UP	U
DOWN	D

F. (AXIS GROUP) ::= (AXIS)b(NUMBER)

G. (AXIS BLOCK) ::= (AXIS BLOCK)b,b(AXIS GROUP) or (AXIS GROUP)

H. (DIRECTION GROUP) ::= (DIRECTION) (NUMBER)

I. (DIRECTION BLOCK) ::= (DIRECTION BLOCK)b,b(DIRECTION GROUP) or (DIRECTION GROUP)

J. (VIEW GROUP) ::= (VIEW GROUP)b,b(VIEW) or (VIEW)

K. (POINT) ::= (NUMBER)b,b(NUMBER)b,b(NUMBER)

The three NUMBERS represent the X, Y, Z coordinates respectively.

L. (POINT GROUP) ::= (POINT)b,b(SHOW-HIDE GROUP) or (POINT)

M. (POINT LIST) ::= (POINT GROUP)b$b(POINT LIST) or

(POINT GROUP)b$

O. (SHOW-HIDE GROUP) ::= (SHOW WORD) (FACE COMB GROUP)

P. (SHOW WORD) A SHOW WORD is one of the following words or abbreviations.

SHOW WORD	ABBREVIATION
SHOW	S
HIDE	H

Q. (FACE COMB GROUP) ::= (FACE COMB GROUP)b,b(FACE COMB) or

(FACE COMB)

R. (FACE COMB) A FACE COMB is taken from the following list:

FACE COMB	ABBREVIATION
FRONT TOP or TOP FRONT	F T or T F
FRONT BOTTOM or BOTTOM FRONT	F BO or BO F
FRONT RIGHT or RIGHT FRONT	F R or R F
FRONT LEFT or LEFT FRONT	F L or L F
BACK TOP or TOP BACK	B T or T B
BACK BOTTOM or BOTTOM BACK	B BO or BO B
BACK RIGHT or RIGHT BACK	B R or R B
BACK LEFT or LEFT BACK	B L or L B
RIGHT TOP or TOP RIGHT	R T or T R
RIGHT BOTTOM or BOTTOM RIGHT	R BO or BO R
LEFT TOP or TOP LEFT	L T or T L
LEFT BOTTOM or BOTTOM LEFT	L BO or BO L
FRONT	F
BACK	B
RIGHT	R
LEFT	L
TOP	T

FACE COMB	ABBREVIATION
BOTTOM	BO
ITSELF	I
ALL	A

The purpose of the SHOW-HIDE GROUP is to define which combinations of faces a line will be seen with in the solid format. A SHOW GROUP defines the face combinations with which the line will be seen. The line will be hidden for all other combinations. Similarly a HIDE GROUP defines the face combinations for which the line will be hidden. The line will be seen for all other combinations of faces. If the FACE COMB "ITSELF" is included in a SHOW-HIDE GROUP, the line will be appropriately shown or hidden if the face the line is defined on is shown in combination with no other faces or, in other words, shown by itself. The FACE COMB "ALL" means that a line will be appropriately shown or hidden for all combinations of faces. If a POINT is not followed by a SHOW-HIDE GROUP, a "SHOW ALL" SHOW-HIDE GROUP is assumed.

For example, a line on the front face with a SHOW-HIDE GROUP like "SH LE, TO R, IT, BA BO" will only be seen anytime the front face is displayed by itself, with both the top and right faces, and with the left face alone. The "BACK BOTTOM" combination will be ignored because it is impossible for the front face to be seen with both the back and bottom faces in a solid format. A "LEFT RIGHT" combination would be treated as if the "RIGHT" had been presented alone.

II. SCOPE INSTRUCTIONS

The SCOPE instructions have been divided into five categories: Storage Manipulation Instructions, Display Format Instructions, Figure Orientation Instructions, Line Control Instructions, and Miscellaneous Instructions. All instructions will be displayed in the following format except for the DEFINE BLOCK:

COMPLETE EXPANDED VERSION

ABBREVIATED VERSION

Following the formal definition of each instruction an explanation of the purpose of the instruction will be presented.

A. STORAGE MANIPULATION INSTRUCTIONS

1. DEFINE BLOCK

The DEFINE BLOCK is a special package of statements that is used to input new figures into storage. A statement is considered to be any block of characters terminated by a dollar sign or the "END," instruction. Any number of statements may be put on a line of input, but any statement started on a line must be completed on that line. If storage limitations are exceeded while a figure is being defined, the entire DEFINE BLOCK must be resubmitted.

The first statement is the DEFINE STATEMENT.

```
DEFINE (NAME)b$
DE (NAME)b$
```

The DEFINE STATEMENT is used to input the NAME of the figure.

Two figures can have the same NAME, and the user will be able to process either one. If two figures do have the same NAME, however, the user must be careful not to erase the wrong one. During manipulation, the figures will change places in storage, and the ERASE instruction erases the first figure of a given NAME that is found.

The next statement is the SCALE STATEMENT.

```
SCALE (NUMBER)b$
S (NUMBER)b$
```

The SCALE is used to define the limits of the axes on the screen. A SCALE of 1000, for example, will define a 1000 by 1000 grid on the screen which means that all X, Y, and Z coordinates must be greater than -500 and less than +500.

Following the SCALE STATEMENT the data points for each face must be provided in the following format:

```
(VIEW)b$b(POINT LIST) END,
(VIEW)b$b(POINT LIST) E,
```

The data points following the last face will be followed by an "END$" STATEMENT instead of the "END," STATEMENT that follows the other faces. The faces must be presented in the following order: FRONT,BACK,RIGHT,LEFT,TOP, and BOTTOM. The BACK, LEFT, or BOTTOM faces can be omitted and they will be generated from the FRONT, RIGHT, or TOP faces respectively. A face is omitted by merely leaving it out of the DEFINE BLOCK entirely. If the BACK face is omitted, it will be generated as the mirror image of the FRONT face reflected about the Y-Z

plane, and each point in the BACK face will have the same SHOW-HIDE GROUP as the corresponding point in the FRONT face. Similarly, if the LEFT face is omitted, it will be generated as the mirror image of the RIGHT face reflected about the X-Z plane, and the generated BOTTOM face will be the mirror image of the TOP face reflected about the X-Y plane. The following DEFINE BLOCK can be used to input a cube. The use of the abbreviations is also demonstrated.

```
        DEFINE CUBE   $
        S 1000$  FRONT$
        500,500,500$ 500,500,-500,SH A $ 500,-500,-500$
        500,-500,  500 $ 500,500,500,  SHOW ALL$
        END,
        RI$ 500,500,500$ 500,500,-500$
        -500,500,-500$ -500,500,500$
        500,500,500$ E, TOP   $
        500,500,500$ 500,-500,500$ -500,-500,500$
        -500,500,500$  500  ,500,500$  EN$
```

2. <u>ERASE INSTRUCTION</u>

```
   ERASE (NAME)b$
   ERA (NAME)b$
```

The ERASE INSTRUCTION is used to remove a figure from storage and free the storage for another figure.

3. <u>INPUT INSTRUCTION</u>

```
   INPUT (NUMBER)b,b(NAME)b$
   INP (NUMBER)b,b(NAME)b$
```

The INPUT INSTRUCTION is used to reload a figure that has been saved with the OUTPUT INSTRUCTION. The NUMBER is the unit number of the input device. The figure will be reassigned the NAME provided with the instruction. If enough storage is not available, the instruction must be resubmitted, and the input device must be reset.

4. LOAD INSTRUCTION

```
LOAD (NAME)b$
LO (NAME)b$
```

The LOAD INSTRUCTION is used display a different figure or to reload the present figure. A figure will be displayed exactly as the user defined it with the front face displayed by itself, The figure will be displayed as a solid, the intensity is set to two, and no face will be dashed.

5. OUTPUT INSTRUCTION

```
OUTPUT (NUMBER)b,b(NAME)b$
O (NUMBER)b,b(NAME)b$
```

The OUTPUT INSTRUCTION is used to save a figure on peripheral storage so that core storage can be freed for other figures. The NUMBER is the unit number of the output device, and the NAME tells which figure to save. It is the user's responsibility to be sure the output device has been readied and has been assigned to the NUMBER.

6. WRITE INSTRUCTION

```
WRITE (NAME)b$
W (NAME)b$
```

The WRITE INSTRUCTION is used to print the internal structure of a figure on the line printer. It is useful as a diagnostic tool.

B. DISPLAY FORMAT INSTRUCTIONS

1. <u>CLEAR INSTRUCTION</u>

```
CLEARb$
Cb$
CLEAR DASHb$
C Db$
CLEAR UNDASHb$
C Ub$
```

The CLEAR INSTRUCTION is used to display the figure in a wire frame format. The "CLEAR$" and "CLEAR DASH$" instructions are used to display the faces that would normally be hidden in the solid format with dashed lines. The "CLEAR UNDASH$" instruction is used to display all faces with solid lines. Those faces that have been declared dashed with the DASH INSTRUCTION are not affected.

2. <u>DASH INSTRUCTION</u>

```
DASH (VIEW GROUP)b$
D (VIEW GROUP)b$
```

The DASH INSTRUCTION is used to cause each of the faces defined in the VIEW GROUP to be displayed with dashed lines. "D F,B,TOP$" is an example of a correct DASH INSTRUCTION.

3. INTENSITY INSTRUCTION

INTENSITY (NUMBER)b$

INT (NUMBER)b$

The INTENSITY INSTRUCTION is used to change the brightness of the figure displayed. A large NUMBER will display a bright figure while a small NUMBER will display a dim figure. The NUMBER must lie between -1024 and +1023.

4. SOLID INSTRUCTION

SOLIDb$

SOb$

The SOLID INSTRUCTION is used to cause the figure to be displayed in a solid format.

5. UNDASH INSTRUCTION

UNDASH (VIEW GROUP)b$

U (VIEW GROUP)b$

The UNDASH INSTRUCTION is the opposite of the DASH INSTRUCTION and causes the faces in the VIEW GROUP to be displayed with solid lines.

C. FIGURE ORIENTATION INSTRUCTIONS

1. FACE ORIENTATION INSTRUCTIONS

(VIEW)b$

The FACE ORIENTATION INSTRUCTIONS are used to cause any face of the figure to be presented by itself as an orthogonal projection exactly as the user defined it.

2. <u>EXPAND INSTRUCTION</u>

```
EXPAND (NUMBER)b$
EX (NUMBER)b$
```

The EXPAND INSTRUCTION is used to enlarge the figure by subtracting the NUMBER from the scale of the figure. A negative NUMBER will cause the figure to become smaller.

3. <u>EXPAND TO INSTRUCTION</u>

```
EXPAND TO (NUMBER)b$
EX T (NUMBER)b$
```

The EXPAND TO INSTRUCTION is used to reset the scale to the NUMBER.

4. <u>MOVE INSTRUCTION</u>

```
MOVE (DIRECTION BLOCK)b$
M (DIRECTION BLOCK)b$
```

The MOVE INSTRUCTION is used to cause the figure to move left, right, up, or down from its present position according to the DIRECTIONS and NUMBERS in the DIRECTION BLOCK. "M L 400, U 500, L 100 $" is a valid MOVE INSTRUCTION. The final translation will be the sum of all the moves defined.

5. <u>MOVE TO INSTRUCTION</u>

```
MOVE TO (DIRECTION BLOCK)b$
M T (DIRECTION BLOCK)b$
```

The MOVE TO INSTRUCTION resets both move indicators to zero which centers the figure; the figure is then translated from that point exactly as if a MOVE INSTRUCTION had been submitted.

6. ROTATE INSTRUCTION

```
ROTATE (AXIS BLOCK)b$

RO (AXIS BLOCK)b$
```

The ROTATE INSTRUCTION is used to rotate the figure around the fixed, reference axes. Each AXIS in the AXIS BLOCK defines the axis around which a rotation is to occur. The NUMBER defines the degrees of rotation and the direction. A positive rotation will appear to be clockwise when observed from the positive end of an axis looking toward the origin. "ROTATE X40, Y 32,Z 66$" is a valid ROTATION INSTRUCTION.

7. ROTATE TO INSTRUCTION

```
ROTATE TO (AXIS BLOCK)b$

RO T (AXIS BLOCK)b$
```

The ROTATE TO INSTRUCTION resets the figure to the orientation originally defined by the user; the figure is then rotated exactly as if a ROTATE INSTRUCTION had been submitted.

8. SHRINK INSTRUCTION

```
SHRINK (NUMBER)b$

SHR (NUMBER)b$
```

The SHRINK INSTRUCTION is the opposite of the EXPAND INSTRUCTION and is used to make the figure smaller by adding the NUMBER to the scale of the figure. A negative NUMBER will naturally enlarge the figure.

9. <u>SHRINK TO INSTRUCTION</u>

```
SHRINK TO (NUMBER)b$
SHR T (NUMBER)b$
```

The SHRINK TO INSTRUCTION does exactly the same operation as the EXPAND TO INSTRUCTION and is used to reset the scale of the figure to the NUMBER.

D. LINE CONTROL INSTRUCTIONS

1. <u>ADD INSTRUCTION</u>

```
ADD (VIEW) (POINT)b,b(POINT GROUP)b$
A (VIEW) (POINT)b,b(POINT GROUP)b$
```

The ADD INSTRUCTION is used to add one line to the face defined by the VIEW with a SHOW-HIDE GROUP to determine which combinations of the faces the line will be seen with. If storage limitations are exceeded, the entire instruction must be resubmitted. "AD FR 500,500,500,600,650,12, SH T L, B L,I$" is a correct ADD INSTRUCTION.

2. <u>HIDE INSTRUCTION</u>

```
HIDE (VIEW) (POINT)b,b(POINT)b$
H (VIEW) (POINT)b,b(POINT)b$
```

The HIDE INSTRUCTION is used to cause the line defined by the POINTS on the face defined by the VIEW to be hidden any time the indicated combination of faces is presented. The line hidden would normally have been displayed in a solid format. The HIDE INSTRUCTION sets the bit in the SHOW-HIDE WORD which corresponds to the indicated combination of faces to zero. The line will not be affected for any other combination of faces.

3. <u>SHOW INSTRUCTION</u>

SHOW (VIEW) (POINT)b,b(POINT)b$

SH (VIEW) (POINT)b,b(POINT)b$

The SHOW INSTRUCTION is the opposite of the HIDE INSTRUCTION. The SHOW INSTRUCTION is used to cause the line defined by the POINTS on the face defined by the VIEW to be shown any time the indicated combination of faces is presented. The line would normally have been hidden in the solid format. The SHOW INSTRUCTION sets the bit in the SHOW-HIDE WORD which corresponds to the indicated combination of faces to one.

4. <u>SUBTRACT INSTRUCTION</u>

SUBTRACT (VIEW) (POINT)b,b(POINT)b$

SU (VIEW) (POINT)b,b(POINT)b$

The SUBTRACT INSTRUCTION is the compliment of the ADD INSTRUCTION, but it is not the exact opposite. The line defined by the POINTS in the face defined by the VIEW is hidden in all orientations of the figure, but the line is not actually removed from storage. All of the bits in the SHOW-HIDE WORD are set to zero.

E. MISCELLANEOUS INSTRUCTIONS

1. <u>DONE INSTRUCTION</u>

DONEb$

DOb$

The DONE INSTRUCTION is issued at the end of the block of instructions from unit 5 to indicate the completion

of that block. The next instruction must be submitted from the terminal.

2. <u>GET INSTRUCTION</u>

    GETb$

    Gb$

The GET INSTRUCTION is used to cause the next input to come from the unit designated as 5. The default case is the card reader. Any sequence of instructions may be submitted from unit 5. To return control to the terminal, the sequence must be followed by the DONE INSTRUCTION. Any errors on input from unit 5 may be corrected at the terminal.

3. <u>STOP INSTRUCTION</u>

    STOPb$

    STOPb$

The STOP INSTRUCTION is used to indicate to the system that the session is completed and the computer is released to the next user.

4. <u>ZAP INSTRUCTION</u>

    ZAPb$

    Zb$

The ZAP INSTRUCTION is used to erase the figure from the screen. Storage is not affected, and the figure will be redisplayed with the next instruction. The purpose of the ZAP INSTRUCTION is to protect the display if an error occurs and a bright line or dot appears that could burn the phosphors on the screen. With this instruction the user will not have to terminate his session to protect the equipment.

APPENDIX B

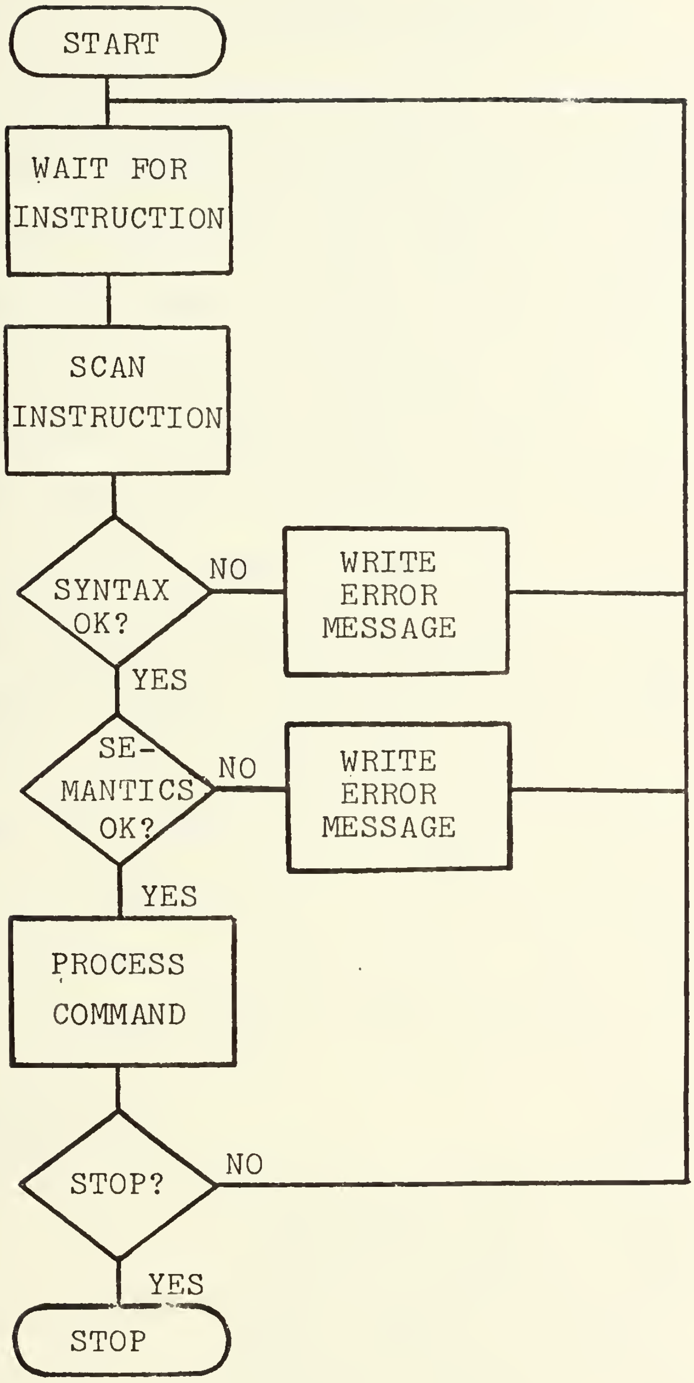
IMPLEMENTATION

The SCOPE language interpreter has been implemented to FORTRAN IV on an XDS 9300 computer interfaced with an Adage AGT 10 graphics display terminal at the Naval Postgraduate School. The system has a graphics display package consisting of seven FORTRAN callable subroutines described in Ref. 11; all FORTRAN instructions used are described in Ref. 8. The basic system has been diagramed on page 39.

The first instruction is expected to come from the display terminal. It is read in as BCD text and scanned to check for validity, syntax, and semantics. If any of these checks fail, an error message is displayed. Only the part of an instruction which is in error need be corrected, and the instruction can then be resubmitted. A check is also made to ensure that enough storage is available for those instructions which require storage allocation. An error message will be displayed if insufficient storage is available, and the instruction will not be accepted.

If the above conditions are all satisfied, the instruction will be processed. If there are errors, error messages will be displayed and the instruction will be rejected. For example, an error message will be generated if an ERASE or LOAD command is given for a figure name that has not been defined. If the instruction requires the figure be displayed,

the entire instruction is processed before the new orientation is displayed. For example, all of the rotations defined by "ROTATE X 45,Y 37,Z 66$" will be applied before the figure is actually displayed. After the instruction is completed, the next instruction can be scanned. If the instruction has not been submitted, the interpreter will go into the wait loop and wait for it.



Basic System Flow Chart

I. STORAGE ALLOCATION

The storage is dynamically allocated and is tightly packed as new figures are defined and old ones are erased. Part of the storage for each figure is a pointer to the beginning of the next figure, and the pointer in the last figure is set to zero. The figure currently being displayed is always moved to the end of storage by the LOAD instruction. This makes the ADD instruction faster than it would normally be if the current figure were at the beginning or in the middle of the storage block because the ADD INSTRUCTION requires storage allocation in the middle of the storage for a figure. All storage below it must be moved down to make room. The purpose of moving a figure to the end of storage is to minimize the amount of storage that must be relocated each time. The storage is all integer and is allocated to each figure in the following format:

RELATIVE LOCATION NUMBER	CONTENTS
0	FIGURE NAME IN A4 FORMAT
1	ABSOLUTE POINTER TO NEXT FIGURE
2	TOTAL LENGTH OF FIGURE
3	ABSOLUTE POINTER TO FRONT FACE STORAGE
4	ABSOLUTE POINTER TO BACK FACE STORAGE
5	ABSOLUTE POINTER TO RIGHT FACE STORAGE
6	ABSOLUTE POINTER TO LEFT FACE STORAGE
7	ABSOLUTE POINTER TO TOP FACE STORAGE
8	ABSOLUTE POINTER TO BOTTOM FACE STORAGE

RELATIVE LOCATION NUMBER	CONTENTS
9	SCALE
10	INTENSITY
11	WIRE FRAME OR SOLID INDICATOR
12-14	DIRECTION COSINES FOR X AXIS
15-17	DIRECTION COSINES FOR Y AXIS
18-20	DIRECTION COSINES FOR Z AXIS
21	HORIZONTAL MOVE INDICATOR
22	VERTICAL MOVE INDICATOR
23	BEGINNING OF FACE STORAGE

The first word of the storage for each face is an indicator which tells whether that face is to be represented with dashed or solid lines. The points are then stored with four words of storage for each point. The first three words contain the X, Y, and Z coordinates respectively, and the fourth word, which is called the SHOW-HIDE word, is an indicator to tell when the line that is ended by that point is to be displayed. Each of the rightmost nineteen bits of the fourth word is an indicator for one of the possible combinations of faces that line can be displayed with; if the bit is a one, the line ended by that point will be seen with that combination of faces. If the line is not to be seen with a given combination, the bit corresponding to that combination will be set to zero. Naturally, the first point for a face is not the end of any lines, and each bit in the fourth word is set to zero to indicate this fact.

II. ROTATION ALGORITHM

Each figure is defined according to a fixed set of axes relative to the screen. There is also a set of axes relative to the figure. When the figure is defined, the relative axes are aligned with the fixed axes, and the direction cosines stored with the figure are the direction cosines of the relative axes in terms of the fixed axes. Rotations are actually applied to the relative axes, and a new set of direction cosines are defined for each axis. The new direction cosines can then be used to rotate the figure without changing the original data points defined by the user. The accuracy of the internal points will therefore not be changed by roundoff error because the internal points are never changed.

It is a very straightforward task to rotate the axes. Each axis is treated as a unit vector from the origin to a point defined by the direction cosines. To rotate the entire set of axes, each individual axis is rotated separately as a unit vector. For example, the unit vector (X,Y,Z) can be rotated about the fixed Z axis to (X1,Y1,Z1) in the following manner.

Because the rotation is about the Z axis, the Z coordinate of the point will not be changed which means that Z1=Z. The X and Y coordinates will change. It is only necessary, however, to rotate the projection of the original vector on the X-Y plane. The projected vector will be defined by the coordinates (X,Y). Further, because each vector is a unit

vector,

$$X^2+Y^2+Z^2 = X1^2+Y1^2+Z1^2,$$

and because $Z1=Z$, it must be the case that:

$$X^2+Y^2 = X1^2+Y1^2$$

Further, the projection of the original vector and the X axis will have an angle β between them. If L1 is the length of the projected vector, X/L1 will define COS β and Y/L1 will define SIN β. If α is the angle of rotation, the angle $\alpha+\beta$ will be the angle between the rotated projection and the X axis.

It can be shown that

$$\text{SIN}(\alpha+\beta) = \text{SIN } \alpha \text{ COS } \beta + \text{COS } \alpha \text{ SIN } \beta \text{ and}$$

$$\text{COS}(\alpha+\beta) = \text{COS } \alpha \text{ COS } \beta - \text{SIN } \alpha \text{ SIN } \beta.$$

Because $X^2+Y^2 = X1^2+Y1^2$, the length of the rotated projection must also be L1. This means that

$$\text{COS}(\alpha+\beta) = X1/L1 \text{ and}$$

$$\text{SIN}(\alpha+\beta) = Y1/L1.$$

But because COS($\alpha+\beta$) and SIN($\alpha+\beta$) can be defined in terms of SIN α, COS α, SIN β, and COS β, it must be the case that

$$X1 = L1(\text{COS } \alpha \text{ COS } \beta - \text{SIN } \alpha \text{ SIN } \beta) \text{ and}$$

$$Y1 = L1(\text{SIN } \alpha \text{ COS } \beta + \text{COS } \alpha \text{ SIN } \beta).$$

Similar rules can be defined for rotations around the X axis and the Y axis.

This algorithm can be applied to the relative axes no matter what position they are in which means that it is possible to keep a record of any previous rotations. It is also straightforward to generate the rotated points for display.

Because the relative axes are aligned with the fixed axes when the figure is originally defined, the points provided by the user will be the correct coordinates in terms of the relative axes. Further, because the position of the figure never changes in terms of the relative axes, the coordinates provided by the user will always be the correct coordinates for the relative axes for all orientations of the relative axes in terms of the fixed axes. A simple application of the formulas to transform rectangular space coordinates will give the coordinates of the points in terms of the fixed axes to display them.

III. FIGURE DISPLAY

To display a figure, each coordinate must be scaled to a number between positive one and negative one: the defined limits of the fixed Y and Z axes on the screen. To do this, each coordinate is first properly rotated, moved and then divided by the scale. The actual figure displayed is an orthogonal projection of the figure on the fixed Y-Z plane. The X coordinate of each point to be displayed is set to zero so that only the Y and Z coordinates are significant. The actual transformation formulas applied to each point are the following:

$$YD = (U1\ X + U2\ Y + U3\ Z + H)/SCALE$$

$$ZD = (V1\ X + V2\ Y + V3\ Z + T)/SCALE$$

(U1,U2,U3) are the direction cosines of the Y axis relative to the figure in terms of the fixed axes.

(V1,V2,V3) are the direction cosines of the Z axis relative to the figure in terms of the fixed axis.

H is the cumulative amount of horizontal move applied to the figure.

T is the cumulative amount of vertical move applied to the figure.

SCALE,X,Y, and Z are provided by the user for each point.

YD is the horizontal coordinate to be displayed.

ZD is the vertical coordinate to be displayed.

After each coordinate is calculated a check is made to determine whether the line should be a move or a draw. This is accomplished by first determining which combination of

faces would be displayed in a solid format. The faces are treated in pairs: Front-Back, Right-Left, and Top-Bottom. The X direction cosine of each of the relative axes is then checked to see which face of the pair will be displayed. The X coordinate is tested because the X axis faces the viewer. If the X coordinate is zero for any axis, neither face in a pair will be displayed. The following chart shows the relationship of the axes and the faces.

	VALUE OF X DIRECTION COSINE		
AXIS	0	+	-
X	Neither	Front	Back
Y	Neither	Right	Left
Z	Neither	Top	Bottom

The SHOW-HIDE word for each point is then tested to determine whether the bit corresponding to that combination of faces is on or off; if the bit is on, the line will be drawn. Finally, if the figure is to be displayed in a wire frame format, all lines will be displayed except those which have all of the bits set to zero in the SHOW-HIDE word.

```
C     MAIN PRØGRAM
      CØMMØN/IFIGUR/IFIGUR(5000),IFP,IFIG
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      CØMMØN/GRAPH/IDEV1,IGRAPH(40)
      CØMMØN/MESSØ/MESSØ(24),IDEV,IDIR(4)
      DIMENSIØN INUMB(6),IVIEW(3),LETTERS(26)
      DATA ICØM,IDØLS,IBLANK,IHIDE/4H,    ,4H$   ,4H    ,07777777B/
      DATA LETTERS/4HA   ,4HB   ,4HC   ,4HD   ,4HE   ,4HF   ,
     1 4HG   ,4HH   ,4HI   ,4HJ   ,4HK   ,4HL   ,4HM   ,4HN   ,4HØ   ,
     2 4HP   ,4HQ   ,4HR   ,4HS   ,4HT   ,4HU   ,4HV   ,4HW   ,4HX   ,
     3 4HY   ,4HZ   /
      NAMELIST IDEV
C     IFP PØINTS TØ THE FIRST ELEMENT IN FREE STØRAGE
C     IFIG PØINTS TØ THE FIGURE BEING PRØCESSED
C     IP PØINTS TØ THE BEGINNING ØF THE PRESENT INSTRUCTIØN
C     INDIC IS A GENERAL PURPØSE PØINTER
C     IDEV TELLS WHICH SCØPE IS BEING USED
C     IDIR TELLS HØW MANY BLØCKS ARE SET UP FØR MESSAGES
C     IGRAPH TELLS HØW MANY BLØCKS ARE SET UP FØR GRAPHICAL ØUTPUT
C
C     INITIALIZATIØN
 1000 IDEV = 1
      ØUTPUT(101) 'IDEV=1*'
      INPUT(101)
      IDEV1 = IDEV
      INMARK = 0
      IFP = 1
      IDEF = 0
      IFIG = 0
      IERR = 0
      CALL DTINIT(IDEV,IDIR,4,IE)
      CALL DGINIT(IDEV1,IGRAPH,IE)
  899 IF(INMARK.EQ.0) GØTØ 896
      READ(5,1001) (INPUT(I),I=1,80)
 1001 FØRMAT(80A1)
```

```
      DØ 895 I=81,96
      INPUT(I) = IBLANK
  895 CØNTINUE
      CALL CLM(0)
      ENCØDE(80,1001,MESS8) (INPUT(I),I=1,80)
      CALL TEXT8(IDEV,MESS8,24,1,1,1,2,IE)
      GØTØ 900
  896 CALL CLM(0)
      CALL TEXT8(IDEV,MESS8,24,1,1,1,2,IE)
      CALL CLM(0)
      ENCØDE(28,2000,MESS8)
 2000 FØRMAT('PLEASE SUBMIT INSTRUCTIØNS  ')
      CALL TEXT8(IDEV,MESS8,24,2,1,1,2,IE)
      IDIR(1) = IDIR(1)-MØD(IDIR(1),8)
  897 IF(MØD(IDIR(1),8)) 900,897,900
  900 CALL CLM(0)
      CALL TEXT8(IDEV,MESS8,24,2,1,1,2,IE)
      CALL TEXT8(IDEV,MESS8,24,3,1,1,2,IE)
      IP = 1
      IPP = IP
      IF(IERR.EQ.1) GØTØ 951
      IF(INMARK.EQ.1) GØTØ 950
C     GET INSTRUCTIØNS FRØM THE SCØPE
  951 CALL CLM(1)
      IERR = 0
  950 IF(IDEF.NE.0) GØTØ (401,410,412,414),IDEF
      CALL SCAN
      IPP = IP
      CALL BLANKS(IPP,0)
      GØTØ(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,
     123,24,25,26,27,28,29,30,31,32,33,34),INDIC
C     ADD INSTRUCTIØN
    1 CALL BLANKS(IPP,1)
      CALL VIEW(IPP)
      IF(INDIC.EQ.0) GØTØ 901
```

```
      CALL BLANKS(IPP,0)
      DØ 100 I=1,6
      IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 102
      INUMB(I) = INT
      IPP = 1+IPP
  100 CØNTINUE
      CALL SHØW(IPP,INT)
      IF(IPP.LT.0) GØTØ 909
      GØTØ 103
  102 IF(I.LT.6) GØTØ 902
      IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      INUMB(6) = INT
      INT = IHIDE
C     PUT THE NEW LINE AT THE BACK ØF THE LIST FØR THE FACE
  103 IFIGUR(IFIG+2) = IFIGUR(IFIG+2)+8
      IF(INDIC.EQ.6) GØTØ 104
      DØ 105 I=INDIC+1,6
      IFIGUR(IFIG+2+I) = IFIGUR(IFIG+2+I)+8
  105 CØNTINUE
  104 NAME = IFIGUR(IFIG+3+INDIC)-8
      IF(INDIC.EQ.6) NAME=IFIG+IFIGUR(IFIG+2)+1
      INFØRM = 3
      CALL STØAL3(NAME,8,INFØRM)
      IF(INFØRM.EQ.0) GØTØ 911
      DØ 106 I=1,3
  106 IFIGUR(NAME+I-1) = INUMB(I)
      IFIGUR(NAME+3) = 0
      DØ 107 I=4,6
  107 IFIGUR(NAME+I) = INUMB(I)
      IFIGUR(NAME+7) = INT
C     CALL THE DISPLAY RØUTINE SECTIØN
```

```
      GØTØ 800
C     BACK
    2 CALL AXRØT(0.,5)
      GØTØ 800
C     BØTTØM
    3 CALL AXRØT(0.,9)
      GØTØ 800
C     DEFINE
    4 CALL CLM(0)
      IDIR(1) = IDIR(1)-MØD(IDIR(1),8)
      CALL TEXTØ(IDEV,MESSØ,24,1,1,1,2,IE)
      ENCØDE(88,1002,MESSØ)
 1002 FØRMAT('IF YØU WANT TØ CØNTINUE WITH A DEFINE BLØCK, TYPE YES ',
     1'IN LINE 1 ... ØTHERWISE TYPE NØ    ')
      CALL TEXTØ(IDEV,MESSØ,24,2,1,1,2,IE)
  402 IF(MØD(IDIR(1),8)) 403,402,403
  403 CALL TEXTI(IDEV,MESSØ,24,1,1,IE)
      DECØDE(24,1003,MESSØ) (MESSØ(I),I=1,24)
 1003 FØRMAT(24A1)
      IDIR(1) = IDIR(1)-MØD(IDIR(1),8)
      DØ 404 I=1,24
      IF(MESSØ(I).EQ.LETTERS(25)) GØTØ 406
  404 CØNTINUE
  406 CALL CLM(0)
      CALL TEXTØ(IDEV,MESSØ,24,1,1,1,2,IE)
      CALL TEXTØ(IDEV,MESSØ,24,2,1,1,2,IE)
      IF(I.LE.24) GØTØ 400
      CALL ERR(IP)
      GØTØ 910
  400 IDEF = 1
C     PACK NAME
  401 CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GØTØ 903
      IFIGUR(IFP) = NAME
      NAME = 0
```

```
      IF(IFIG.EQ.0) GOTO 405
      IFIGJR(IFIG+1) = IFP
  405 IFIG = IFP
      IFIGJR(IFIG+1) = 0
      IHOLD = 22+IFIG
      IF(IHOLD.GE.5000) GOTO 911
      CALL BLANKS(IPP,0)
      CALL BLANKS(IPP,1)
      IDEF = 2
      IF(IPP.NE.96) GOTO 411
      GOTO 899
  410 IPP = 1
  411 CALL BLANKS(IPP,1)
      IP = IPP
      CALL BLANKS(IPP,0)
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GOTO 902
      IFIGJR(IFIG+9) = INT
  421 IPP = IPP+1
      CALL BLANKS(IPP,1)
      IDEF = 3
      IF(IPP.NE.96) GOTO 413
      GOTO 899
  412 IPP = 1
  413 CALL BLANKS(IPP,1)
      IP = IPP
      CALL VIEW(IPP)
      IF(INDIC.EQ.0) GOTO 901
      IHOLD = IHOLD+1
      IF(IHOLD.GE.5000) GOTO 911
      IFIGJR(IFIG+2+INDIC) = IHOLD
  422 IF(INDIC.NE.NAME+1) GOTO 435
      CALL BLANKS(IPP,0)
      IF(INPUT(IPP-1).EQ.IDOLS) GOTO 420
      CALL BLANKS(IPP,1)
```

```
      IPP = IPP+1
  420 CALL BLANKS(IPP,1)
      INFØRM = 2
      IDEF = 4
      IF(IPP.NE.96) GØTØ 415
      GØTØ 899
  414 IPP = 1
  415 CALL BLANKS(IPP,1)
      IP = IPP
      IF(INPUT(IPP).EQ.LETTERS(5)) GØTØ 430
      DØ 416 I=1,3
      IFP = IPP-1
  424 IFP = IFP+1
      CALL BLANKS(IFP,INFØRM)
      IF(IFP.GT.0) GØTØ 424
      IF(INFØRM.LT.0) GØTØ 417
      IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      INUMB(I) = INT
      IPP = IPP+1
  416 CØNTINUE
      CALL SHØW(IPP,INT)
      IF(IPP.LT.0) GØTØ 909
      IPP = IPP+1
      GØTØ 418
  417 IF(I.LT.3) GØTØ 902
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      IPP = IPP+1
      INUMB(3) = INT
      INT = IHIDE
  418 DØ 419 I=1,3
      IFIGUR(IHØLD+I) = INUMB(I)
  419 CØNTINUE
```

```
      IFIGUR(IHØLD+4) = INT
      IHØLD = IHØLD+4
      IF(IHØLD.GE.5000) GØTØ 911
      GØTØ 420
  430 CALL BLANKS(IPP,0)
      IPP = IPP-1
      IF(INPUT(IPP).EQ.IDØLS) GØTØ 450
      IF(INPUT(IPP).EQ.ICØM) GØTØ 431
      IPP = IPP+1
      CALL BLANKS(IPP,1)
C     DØNE WITH GRØUP
      IF(INPUT(IPP).EQ.IDØLS) GØTØ 450
  431 NAME = INDIC
      GØTØ 421
  435 IFIGUR(IFIG+3+NAME) = IHØLD
      DØ 436 I=IFIGUR(IFIG+2+NAME)+1,IFIGUR(IFIG+3+NAME)-1
      IHØLD = IHØLD+1
      IF(IHØLD.GE.5000) GØTØ 911
      IFIGUR(IHØLD) = IFIGUR(I)
  436 CØNTINUE
      DØ 437 I=IFIGUR(IFIG+3+NAME)+1+NAME/2,IHØLD,4
      IFIGUR(I) = -IFIGUR(I)
  437 CØNTINUE
      NAME = NAME+1
      IHØLD = IHØLD+1
      IF(IHØLD.GE.5000) GØTØ 911
      IFIGUR(IFIG+2+INDIC) = IHØLD
      GØTØ 422
  450 IF(INDIC.EQ.6) GØTØ 460
      IHØLD = IHØLD+1
      IF(IHØLD.GE.5000) GØTØ 911
      IFIGUR(IFIG+8) = IHØLD
      DØ 451 I=IFIGUR(IFIG+7)+1,IFIGUR(IFIG+8)-1
      IHØLD = IHØLD+1
      IF(IHØLD.GE.5000) GØTØ 911
```

```
      IFIGUR(IHOLD) = IFIGUR(I)
  451 CONTINUE
      DO 452 I=IFIGUR(IFIG+8)+3,IHOLD,4
      IFIGUR(I) = -IFIGUR(I)
  452 CONTINUE
  460 IFP = IHOLD+1
      DO 465 I=1,6
      IFIGUR(IFIGUR(IFIG+2+I)+4) = 0
  465 CONTINUE
      IFIGUR(IFIG+2) = IFP-IFIG
      IDEF = 0
      GOTO 124
C     EXPAND
    5 CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GOTO 902
      IF(INDIC.EQ.6) INT=-INT
   51 IFIGUR(IFIG+9) = IFIGUR(IFIG+9)-INT
      GOTO 800
C     EXPAND TO
    6 CALL BLANKS(IPP,1)
      CALL BLANKS(IPP,0)
      IFIGUR(IFIG+9) = 0
      GOTO 5
C     ERASE
    7 CALL BLANKS(IPP,1)
      IHOLD = IPP
      CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GOTO 903
      I = 1
      IF(IFIG.EQ.0) GOTO 904
   71 IF(IFIGUR(I).EQ.NAME) GOTO 70
      NEXT = I
      I = IFIGUR(I+1)
      IF(I.EQ.0) GOTO 904
      GOTO 71
```

```
   70 IF(IFIGUR(I+1).EQ.0) GØTØ 75
      NEXT = IFIGUR(I+1)
   72 DØ 74 J=3,8
      IFIGUR(NEXT+J) = IFIGUR(NEXT+J)-IFIGUR(I+2)
   74 CØNTINUE
      IF(IFIGUR(NEXT+1).EQ.0) GØTØ 73
      IFIGUR(NEXT+1) = IFIGUR(NEXT+1)-IFIGUR(I+2)
      NEXT = IFIGUR(NEXT+1)+IFIGUR(I+2)
      GØTØ 72
   75 IFP = IFIG
      IF(IFIG.EQ.1) NEXT=1
      IFIGUR(NEXT+1) = 0
      IFIG = NEXT
      NAME = 0
      GØTØ 76
   73 IFIG = IFIG-IFIGUR(I+2)
      CALL STØALØ(I,IFIGUR(I+2),2)
   76 IF(IFP.LT.20) IFIG=0
      IF(NAME.NE.0) GØTØ 800
C     GØTØ DISPLAY SECTIØN AND CLEAN ØFF SCREEN IF PRESENT FIG IS ERASED
      GØTØ 31
C     FRØNT
    8 CALL AXRØT(0.,4)
      GØTØ 800
C     GET
    9 INMARK = 1
      GØTØ 899
C     HIDE
   10 NAME = -1
      ANGLE = 1
  140 CALL BLANKS(IPP,1)
      CALL VIEW(IPP)
      IF(INDIC.EQ.0) GØTØ 901
      IEND = IFIGUR(IFIG+INDIC+3)-1
      IF(INDIC.EQ.6) IEND=IFIG+IFIGUR(IFIG+2)-1
```

```
      CALL BLANKS(IPP,0)
      DØ 145 I=1,5
      IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      INUMB(I) = INT
      IPP = IPP+1
  145 CØNTINUE
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      INUMB(6) = INT
C     FIND LINE
      DØ 150 I=IFIGUR(IFIG+2+INDIC)+1,IEND,4
      IF(IFIGUR(I).NE.INUMB(1)) GØTØ 150
      IF(IFIGUR(I+1).NE.INUMB(2)) GØTØ 150
      IF(IFIGUR(I+2).NE.INUMB(3)) GØTØ 150
      IF(IFIGUR(I+4).NE.INUMB(4)) GØTØ 150
      IF(IFIGUR(I+5).NE.INUMB(5)) GØTØ 150
      IF(IFIGUR(I+6).EQ.INUMB(6)) GØTØ 151
  150 CØNTINUE
C     LINE DØES NØT EXIST
      GØTØ 905
  151 CALL SETVEW(IVIEW,IFIGUR(I+7),INDIC,NAME)
      IF(ANGLE.EQ.0) IFIGUR(I+7)=0
      GØTØ 800
C     LEFT
   11 CALL AXRØT(0.,6).
      GØTØ 800
C     LØAD
   12 CALL BLANKS(IPP,1)
      IHØLD = IPP
      CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GØTØ 903
      I = 1
      IF(IFIG.EQ.0) GØTØ 908
```

```
  121 IF(IFIGUR(I).EQ.NAME) GØTØ 122
      I = IFIGUR(I+1)
      IF(I.EQ.0) GØTØ 908
      GØTØ 121
  122 INFØRM = 1
      IFIG = I
      CALL STØALØ(I,IFIGUR(I+2),INFØRM)
C     SET INTENSITY
  124 IFIGUR(IFIG+10) = 2
C     SET DASH WØRDS TØ SØLID LINES
      DØ 123 I=1,6
      IFIGUR(IFIGUR(IFIG+2+I)) = 0
  123 CØNTINUE
C     SET CLEAR-SØLID INDICATØR TØ SØLID
      IFIGUR(IFIG+11) = 0
C     SET MØVE INDICATØRS TØ 0
      IFIGUR(IFIG+21) = 0
      IFIGUR(IFIG+22) = 0
      GØTØ 8
C     MØVE
   13 INFØRM = 2
      IPP = IPP-1
  136 NAME = 1
      IPP = IPP+1
      CALL BLANKS(IPP,1)
      IF(INPUT(IPP).NE.LETTERS(12)) GØTØ 131
      INDIC = -1
      GØTØ 135
  131 IF(INPUT(IPP).NE.LETTERS(18)) GØTØ 132
      INDIC = 1
      GØTØ 135
  132 NAME = 2
      IF(INPUT(IPP).NE.LETTERS(21)) GØTØ 133
      INDIC = 1
      GØTØ 135
```

```
C     DIRECTIØN DØES NØT APPEAR
  133 IF(INPUT(IPP).NE.LETTERS(4)) GØTØ 906
      INDIC = -1
  135 CALL BLANKS(IPP,0)
      IHØLD = IPP-1
  134 IHØLD = IHØLD+1
      CALL BLANKS(IHØLD,INFØRM)
      IF(IHØLD.GT.0) GØTØ 134
      IF(INFØRM.GT.0) GØTØ 137
C     LØØK FØR A DØLLAR SIGN
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      INUMB(3) = 0
      GØTØ 138
  137 INUMB(3) = 1
      IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
  138 IFIGUR(IFIG+20+NAME) = INT*INDIC+IFIGUR(IFIG+20+NAME)
      IF(INUMB(3).EQ.1) GØTØ 136
      GØTØ 800
C     MØVE TØ
   14 CALL BLANKS(IPP,1)
      CALL BLANKS(IPP,0)
      CALL BLANKS(IPP,1)
      IFIGUR(IFIG+21) = 0
      IFIGUR(IFIG+22) = 0
      GØTØ 13
C     RIGHT
   15 CALL AXRØT(0.,7)
      GØTØ 800
C     RØTATE
   16 CALL BLANKS(IPP,1)
      INFØRM = 2
  164 DØ 160 I=1,3
```

```
      IF(INPUT(IPP).EQ.LETTERS(23+I)) GØTØ 162
  160 CØNTINUE
C     IMPRØPER AXIS NAME
      GØTØ 906
  162 IHØLD = IPP
  165 IHØLD = IHØLD+1
      CALL BLANKS(IHØLD,INFØRM)
      IF(IHØLD.GT.0) GØTØ 165
      IF(INFØRM.LT.0) GØTØ 166
C     LØØK FØR A CØMMA
      IPP = -IPP-1
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.GT.0) GØTØ 163
      GØTØ 902
  166 IPP = IPP+1
C     LØØK FØR A $
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      ANGLE = INT
      CALL AXRØT(ANGLE,I)
C     GØTØ DISPLAY SECTIØN
      GØTØ 800
  163 ANGLE = INT
      CALL AXRØT(ANGLE,I)
      IPP = IPP+1
      CALL BLANKS(IPP,1)
      GØTØ 164
C     RØTATE TØ
   17 CALL BLANKS(IPP,1)
      CALL BLANKS(IPP,0)
      CALL AXRØT(0.,4)
      GØTØ 16
C     SHRINK
   18 CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
```

```
      INT = -INT
      GØTØ 51
C     SHRINK TØ
   19 CALL BLANKS(IPP,1)
      CALL BLANKS(IPP,0)
      IFIGUR(IFIG+9) = 0
      GØTØ 18
C     SHØW
   20 NAME = 1
      ANGLE = 1.0
      GØTØ 140
C     SUBTRACT
   21 ANGLE = 0.0
      GØTØ 140
C     TØP
   22 CALL AXRØT(0.,8)
      GØTØ 800
C     INTENSITY
   23 CALL BLANKS(IPP,1)
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      IF(INT.LT.0) INT = -INT
      IFIGUR(IFIG+10) = INT
      GØTØ 800
C     STØP
C     CLEAR THE SCØPE AND STØP
   24 CALL GRAPHØ(IDEV1,INPUT,51,0,IE)
      STØP
C     END ØF THE CURRENT INSTRUCTIØN LIST
   25 GØTØ 899
C     CLEAR
   26 IFIGUR(IFIG+11) = 1
      IF(INPUT(IPP-1).EQ.IDØLS) GØTØ 800
      CALL BLANKS(IPP,1)
      IF(IPP.GE.96) GØTØ 800
```

```
      IF(INPUT(IPP).EQ.LETTERS(21)) IFIGUR(IFIG+11)=-1
      GØTØ 800
C     DASH
   27 INT = 1
      INFØRM = 2
  271 CALL BLANKS(IPP,1)
      CALL VIEW(IPP)
      IF(INDIC.EQ.0) GØTØ 901
      IFIGUR(IFIGUR(IFIG+2+INDIC)) = INT
  272 CALL BLANKS(IPP,INFØRM)
      IF(IPP.LT.0) GØTØ 273
      IPP = IPP+1
      GØTØ 272
  273 IF(INFØRM.LT.0) GØTØ 800
      IPP = -IPP+1
      GØTØ 271
C     DØNE
   28 INMARK = 0
      IP = 96
      GØTØ 899
C     SØLID
   29 IFIGUR(IFIG+11) = 0
      GØTØ 800
C     UNDASH
   30 INT = 0
      INFØRM = 2
      GØTØ 271
C     ZAP
   31 CALL DISPLAY(IVIEW,1)
      GØTØ 801
C     INPUT
   32 IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      IPP = IPP+1
```

```
      CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GØTØ 903
      READ(INT) IHØLD
      IF(IHØLD+IFP.GE.5000) GØTØ 911
      IF(IFIG.EQ.0) GØTØ 322
      IFIGUR(IFIG+1) = IFP
  322 IFIG = IFP
      IFP = IFP+IHØLD
      IFIGUR(IFIG) = NAME
      IFIGUR(IFIG+1) = 0
      IFIGUR(IFIG+2) = IHØLD
      READ(INT) (IFIGUR(IFIG+I),I=3,9)
      DØ 321 I=3,8
      IFIGUR(IFIG+I) = IFIGUR(IFIG+I)+IFIG
  321 CØNTINUE
      READ(INT) (IFIGUR(I),I=23+IFIG,IFP-1)
      GØTØ 124
C     ØUTPUT
   33 IPP = -IPP
      CALL NUMBER(IPP,INT,96,0)
      IF(IPP.LT.0) GØTØ 902
      IPP = IPP+1
      IHØLD = IPP
      CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GØTØ 903
      I = 1
      IF(IFIG.EQ.0) GØTØ 915
  331 IF(IFIGUR(I).EQ.NAME) GØTØ 332
      I = IFIGUR(I+1)
      IF(I.EQ.0) GØTØ 915
      GØTØ 331
  332 INFØRM = IFIGUR(I+1)
      IF(INFØRM.EQ.0) INFØRM=IFP
      WRITE(INT) IFIGUR(I+2)
      DØ 333 J=3,8
```

```
      IFIGUR(I+J) = IFIGUR(I+J)-I
  333 CONTINUE
      WRITE(INT) (IFIGUR(I+J),J=3,9)
      DO 334 J=3,8
      IFIGUR(I+J) = IFIGUR(I+J)+I
  334 CONTINUE
      WRITE(INT) (IFIGUR(J),J=I+23,INFORM)
      GOTO 801
C     WRITE
   34 CALL BLANKS(IPP,1)
      IHOLD = IPP
      CALL NUMBER(IPP,NAME,96,1)
      IF(IPP.LT.0) GOTO 903
      I = 1
      IF(IFIG.EQ.0) GOTO 914
  341 IF(IFIGUR(I).EQ.NAME) GOTO 342
      I = IFIGUR(I+1)
      IF(I.EQ.0) GOTO 914
      GOTO 341
  342 CALL WRITE(I)
      GOTO 801
C     DISPLAY SECTION
  800 CALL DISPLAY(IVIEW,0)
  801 IF(INPUT(IP).EQ.IDOLS) GOTO 802
      IF(IP.EQ.95) GOTO 802
      IP = IP+1
      GOTO 801
  802 IP = IP+1
      GOTO 950
  901 CALL ERR(IPP)
      ENCODE(16,2001,MESSO)
 2001 FORMAT('IMPROPER VIEW    ')
  910 CALLTEXTO(IDEV,MESSO,24,3,1,1,2,IE)
      IERR = 1
      GOTO 897
```

```
 902 IF(INT.NE.0) G0T0 907
     CALL ERR(-IPP)
     ENC0DE(36,2002,MESS0)
2002 F0RMAT('IMPR0PER CHARACTER F0LL0WING NUMBER ')
     G0T0 910
 903 CALL ERR(NAME)
     ENC0DE(52,2004,MESS0)
2004 F0RMAT('THE EXPECTED F0LL0WING CHARACTER 0CCURS IN THE NAME ')
     G0T0 910
 904 CALL ERR(IH0LD)
     ENC0DE(36,2005,MESS0)
2005 F0RMAT('THE FIGURE T0 ERASE D0ES N0T EXIST  ')
     G0T0 910
 905 CALL ERR(IPP)
     ENC0DE(36,2006,MESS0)
2006 F0RMAT('THE LINE D0ES N0T EXIST IN THIS VIEW')
     G0T0 910
 906 CALL ERR(IPP)
     ENC0DE(36,2007,MESS0)
2007 F0RMAT('IMPR0PER AXIS NAME 0R M0VE DIRECTI0N')
     G0T0 910
 907 CALL ERR(INT)
     ENC0DE(32,2003,MESS0)
2003 F0RMAT('IMPR0PER CHARACTER IN A NUMBER  ')
     G0T0 910
 908 CALL ERR(IH0LD)
     ENC0DE(32,2008,MESS0)
2008 F0RMAT('FIGURE T0 L0AD D0ES N0T EXIST   ')
     G0T0 910
 909 CALL ERR(-IPP)
     ENC0DE(28,2009,MESS0)
2009 F0RMAT('IMPR0PER SH0W 0R HIDE GR0UP ')
     G0T0 910
 911 CALL ERR(IP)
     ENC0DE(96,2010,MESS0)
```

```
2010 FØRMAT('STØRAGE LIMITATIØNS HAVE BEEN EXCEEDED ••• PLEASE ERASE ',
    1'A FIGURE AND RESUBMIT ENTIRE INSTRUCTIØN')
     IF(IDEF•EQ•0) GØTØ 910
     I = 1
 912 IF(IFIGUR(I+1)•EQ•IFIG) GØTØ 913
     I = IFIGUR(I+1)
     GØTØ 912
 913 IFIG = I
     IFP = IFIGUR(IFIG+1)
     IFIGUR(IFIG+1) = 0
     IDEF = 0
     GØTØ 910
 914 CALL ERR(IHØLD)
     ENCØDE(32,2011,MESS9)
2011 FØRMAT('FIGURE TØ WRITE DØES NØT EXIST  ')
     GØTØ 910
 915 CALL ERR(IHØLD)
     ENCØDE(32,2012,MESS9)
2012 FØRMAT('FIGURE TØ ØUTPUT DØES NØT EXIST ')
     GØTØ 910
     END
```

```
      SUBRØUTINE AXRØT(BNGLE,JAXIS)
C     THIS SUBRØUTINE RØTATES THE AXIS FØR FIGURE RØTATIØN
C     THE DIRECTIØN·CØSINES FØR EACH FIGURE ARE STØRED IN LØCATIØNS
C     12 - 20 FØR THAT FIGURE
C     THEY ARE CØPIED AS FLØATING PØINT NUMBERS FØR CALCULATIØNS
      CØMMØN/IFIGUR/IFIGUR(5000),IFP,IFIG
      DIMENSIØN DIR(9)
      ANGLE = BNGLE
      IAXIS = JAXIS
      DØ 50 I=1,9
   50 DIR(I) = IFIGUR(IFIG+11+I)/8388607.
C     ANGLE IS SET TØ RADIANS
C     IF A SPECIAL RØTATE INSTRUCTIØN IS INDICATED GØTØ 4
      IF(4-IAXIS) 4,4,8
    8 ANGLE = -6.2832*ANGLE/360.
C     ANGLE WILL ALWAYS BE PØSITIVE
      IF(ANGLE.LT.0) ANGLE = 6.2832+ANGLE
      I = -3
      CØSB = CØS(ANGLE)
      SINB = SIN(ANGLE)
      GØTØ (1,2,3),IAXIS
C     RØTATE ABØUT X
C     PØSITIVE TAKES Z AXIS INTØ Y AXIS
    1 I = I+3
      J = 2+I
      K = 3+I
      GØTØ 5
   11 IF(I-6) 1,6,6
C     RØTATE ABØUT Y
C     PØSITIVE·TAKES X AXIS INTØ Z AXIS
    2 I = I+3
      J = I+3
      K = I+1
      GØTØ 5
   12 IF(I-6) 2,6,6
```

```
C     RØTATE ABØUT Z
C     PØSITIVE TAKES Y AXIS INTØ X AXIS
    3 I = I+3
      J = I+1
      K = I+2
      GØTØ 5
   13 IF(I-6) 3,6,6
    5 ALEN = SQRT(DIR(J)*DIR(J)+DIR(K)*DIR(K))
      IF(ALEN.EQ.0) GØTØ 9
      CØSA = DIR(J)/ALEN
      SINA = DIR(K)/ALEN
    9 DIR(K) = ALEN*(SINA*CØSB+SINB*CØSA)
      DIR(J) = ALEN*(CØSA*CØSB-SINA*SINB)
      GØTØ (11,12,13),IAXIS
    6 DØ 7 I=1,9
    7 IFIGUR(IFIG+11+I) = DIR(I)*8388607.
      RETURN
C     THIS SECTIØN IS FØR THE SPECIAL INSTRUCTIØNS
    4 DØ 10 I=1,9
   10 DIR(I) = 0.0
      IAXIS = IAXIS-3
      GØTØ (21,22,23,24,25,26),IAXIS
C     FRØNT
   21 DIR(1) = 1.0
      DIR(5) = 1.0
      DIR(9) = 1.0
      GØTØ 6
C     BACK
   22 DIR(1) = -1.0
      DIR(5) = -1.0
      DIR(9) = 1.0
      GØTØ 6
C     LEFT
   23 DIR(2) = 1.0
      DIR(4) = -1.0
```

```
      DIR(9) = 1.0
      GØTØ 6
C     RIGHT
   24 DIR(2) = -1.0
      DIR(4) = 1.0
      DIR(9) = 1.0
      GØTØ 6
C     TØP
   25 DIR(3) = -1.0
      DIR(5) = 1.0
      DIR(7) = 1.0
      GØTØ 6
C     BØTTØM
   26 DIR(3) = 1.0
      DIR(5) = 1.0
      DIR(7) = -1.0
      GØTØ 6
      END
```

```
      SUBRØUTINE BLANKS(IPP,INFØRM)
C     THIS RØUTINE LØØKS FØR THE FIRST BLANK ØR THE FIRST NØNBLANK IN A
C     STRING
C     IT ALSØ LØØKS FØR CØMMAS AND DØLLAR SIGNS IF INFØRM = 2
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      DATA IBLANK,ICØM,IDØLS/4H    ,4H,   ,4H$   /
      IF(INFØRM.GE.1) GØTØ 5
C     LØØK FØR A BLANK
    1 IF(INPUT(IPP).EQ.IBLANK) RETURN
      IF(IPP.EQ.96) RETURN
      IPP = IPP+1
      GØTØ 1
C     LØØK FØR THE FIRST NØNBLANK
    5 IF(INFØRM.EQ.2) GØTØ 10
    6 IF(INPUT(IPP).NE.IBLANK) RETURN
      IF(IPP.EQ.96) RETURN
      IPP = IPP+1
      GØTØ 6
   10 IF(INPUT(IPP).EQ.IBLANK) RETURN
      IF(INPUT(IPP).NE.ICØM) GØTØ 11
      IPP = -IPP
      RETURN
   11 IF(INPUT(IPP).EQ.IDØLS) GØTØ 12
      IPP = IPP+1
      GØTØ 10
   12 IPP = -IPP
      INFØRM = -2
      RETURN
      END
```

```
      SUBR8UTINE CLM(IN)
      C8MM8N/MESS8/MESS8(24),IDEV,IDIR(4)
C     THIS R8UTINE CLEARS MESSAGE BUFFERS AND ALS8 READS IN BUFFERS AND
C     DEC8DES THEM
      C8MM8N/INPUT/INPUT(96),IP,INDIC
      DATA IBLANK,NULL/4H    ,77777777B/
      IF(IN.NE.0) G8T8 50
      D8 10 I=1,24
      MESS8(I) = NULL
   10 C8NTINUE
      RETURN
   50 CALL TEXTI(IDEV,MESS8,24,1,1,IE)
      DEC8DE(96,100,MESS8) (INPUT(I),I=1,96)
  100 F8RMAT(96A1)
      RETURN
      END
```

```
      SUBRØUTINE DISPLAY(IVIEW,INFØRM)
C     THIS SUBRØUTINE TAKES THE PØINTS, APPLIES MØVES AND RØTATIØNS, AND
C     DEFINES THE MØVE-DRAW MATRIX TØ ØUTPUT THE FIGURE
C     IT ALSØ FILLS THE IVIEW MATRIX TØ TELL WHICH VIEWS ARE FØRWARD
      CØMMØN/IFIGUR/IFIGUR(5000),IFP,IFIG
      CØMMØN/GRAPH/IDEV1,IGRAPH(40)
      DIMENSIØN DCØS(9),Y(50),Z(50),IDRAW(50),IVIEW(3),IM(51)
      IBLØCK = 1
      INTEN = 2*IFIGUR(IFIG+10)
      SCALE = IFIGUR(IFIG+9)
      IF(INFØRM.EQ.1) GØTØ 61
      RMØVE = IFIGUR(IFIG+21)
      UMØVE = IFIGUR(IFIG+22)
      ICLR = 0
      IF(IFIGUR(IFIG+11).NE.0) ICLR=IFIGUR(IFIG+11)
C     USE REAL DIRECTIØN CØSINES
      DØ 1 I=1,9
      DCØS(I) = IFIGUR(IFIG+11+I)/8388607.
    1 CØNTINUE
C     SET UP THE IVIEW MATRIX
C     I       IVIEW(I) = 0        IVIEW(I) = 1        IVIEW(I) = 2
C     1       NEITHER             FRØNT               BACK
C     2       NEITHER             RIGHT               LEFT
C     3       NEITHER             TØP                 BØTTØM
      J = 0
      DØ 6 I=1,8,3
      J = J+1
      IF(DCØS(I)) 3,4,5
    3 IVIEW(J) = 2
      GØTØ 6
    4 IVIEW(J) = 0
      GØTØ 6
    5 IVIEW(J) = 1
    6 CØNTINUE
      IF(ICLR.EQ.0) GØTØ 30
```

```
C     SHØW ALL LINES IN THE ENTIRE FIGURE
      DØ 29 I=1,6
C     ASSUME THE BACK SIDE IS TØ BE DASHED
      IDASH = 1
      IF(ICLR.EQ.-1) IDASH=IFIGUR(IFIGUR(IFIG+2+I))
      IF(IVIEW((I+1)/2).EQ.I/(I-I/2)) IDASH=IFIGUR(IFIGUR(IFIG+2+I))
      IBEG = IFIGUR(IFIG+2+I)
      IFØLL = IFIGUR(IFIG+3+I)
      IF(I.EQ.6) IFØLL = IFIG+IFIGUR(IFIG+2)
      K = 0
      Y(1) = 0.0
      Z(1) = 0.0
      GØTØ 19
   18 Y(1) = Y(50)
      Z(1) = Z(50)
   19 IDRAW(1) = 0
      K = K+4
      DØ 20 J=2,50
      Y(J) = (DCØS(2)*IFIGUR(IBEG+1)+DCØS(5)*IFIGUR(IBEG+2)+
     1      DCØS(8)*IFIGUR(IBEG+3)+RMØVE)/SCALE
      Z(J) = (DCØS(3)*IFIGUR(IBEG+1)+DCØS(6)*IFIGUR(IBEG+2)+
     1      DCØS(9)*IFIGUR(IBEG+3)+UMØVE)/SCALE
      IDRAW(J) = 1
      IBEG = IBEG+4
      IF(IFIGUR(IBEG).EQ.0) IDRAW(J)=0
      IF(IBEG+1.EQ.IFØLL) GØTØ 25
   20 CØNTINUE
      IF(K.EQ.4) IDRAW(2) = 0
      CALL RTØI(50,Y,Z,IDRAW,IM,IDASH,INTEN)
      IM(1) = IM(1)-MØD(IM(1),4096)+INTEN
      CALL GRAPHØ(IDEV1,IM,51,IBLØCK,IE)
      IBLØCK = IBLØCK+1
      IF(IBLØCK.EQ.40) RETURN
      GØTØ 18
C     FILL ØUT THE PRESENT X, Y, AND IDRAW VECTØRS WITH ZERØES BECAUSE A
```

```
C     VIEW IS COMPLETED
   25 DO 26 L=J+1,50
      Y(L) = 0
      Z(L) = 0
      IDRAW(L) = 0
   26 CONTINUE
      IF(K.EQ.4) IDRAW(2)=0
      CALL RTOI(50,Y,Z,IDRAW,IM,IDASH,INTEN)
      IM(1) = IM(1)-MOD(IM(1),4096)+INTEN
      CALL GRAPH3(IDEV1,IM,51,IBLOCK,IE)
      IBLOCK = IBLOCK+1
      IF(IBLOCK.EQ.40) RETURN
   29 CONTINUE
C     TREAT THE FIGURE AS A SOLID
   30 DO 60 I=1,3
      IF(IVIEW(I).EQ.0) GOTO 60
      KEY = 0
      CALL SETVEW(IVIEW,KEY,I,1)
      IBEG = IFIGUR(IFIG+I+I+IVIEW(I))
      IFOLL = IFIGUR(IFIG+1+I+I+IVIEW(I))
      IF(I+I+IVIEW(I).EQ.8) IFOLL=IFIG+IFIGUR(IFIG+2)
      Y(1) = 0.0
      Z(1) = 0.0
      GOTO 32
   31 Y(1) = Y(50)
      Z(1) = Z(50)
   32 IDRAW(1) = 0
      DO 40 J=2,50
      Y(J) = (DCOS(2)*IFIGUR(IBEG+1)+DCOS(5)*IFIGUR(IBEG+2)+
     1      DCOS(8)*IFIGUR(IBEG+3)+RMOVE)/SCALE
      Z(J) = (DCOS(3)*IFIGUR(IBEG+1)+DCOS(6)*IFIGUR(IBEG+2)+
     1      DCOS(9)*IFIGUR(IBEG+3)+UMOVE)/SCALE
      ICK = IFIGUR(IBEG+4)/KEY
      JCK = ICK/2
      JCK = JCK+JCK
```

```
      IDRAW(J) = 0
      IF(JCK.NE.ICK) IDRAW(J)=1
      IBEG = IBEG+4
      IF(IBEG+1.EQ.IFØLL) GØTØ 45
   40 CØNTINUE
      CALL RTØI(50,Y,Z,IDRAW,IM,IFIGUR(IFIGUR(IFIG+I+I+IVIEW(I))),INTEN)
      IM(1) = IM(1)-MØD(IM(1),4096)+INTEN
      CALL GRAPHØ(IDEV1,IM,51,IBLØCK,IE)
      IBLØCK = IBLØCK+1
      IF(IBLØCK.EQ.40) RETURN
      GØTØ 31
   45 DØ 46 L=J+1,50
      Y(L) = 0.0
      Z(L) = 0.0
      IDRAW(L) = 0
   46 CØNTINUE
      CALL RTØI(50,Y,Z,IDRAW,IM,IFIGUR(IFIGUR(IFIG+I+I+IVIEW(I))),INTEN)
      IM(1) = IM(1)-MØD(IM(1),4096)+INTEN
      CALL GRAPHØ(IDEV1,IM,51,IBLØCK,IE)
      IBLØCK = IBLØCK+1
      IF(IBLØCK.EQ.40) RETURN
   60 CØNTINUE
   61 DØ 62 J=1,50
      Y(J) = 0.0
      Z(J) = 0.0
      IDRAW(J) = 0
   62 CØNTINUE
      CALL RTØI(50,Y,Z,IDRAW,IM,IFIGUR(IFIGUR(IFIG+2+I)),INTEN)
   63 CALL GRAPHØ(IDEV1,IM,51,IBLØCK,IE)
      IBLØCK = IBLØCK+1
      IF(IBLØCK.EQ.40) RETURN
      GØTØ 63
      END
```

```
      SUBRØUTINE ERR(IPP)
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      CØMMØN/MESSØ/MESSØ(24),IDEV,IDIR(4)
      DATA NULL/77777777B/
C     THIS RØUTINE IS CALLED ANY TIME THERE IS AN ILLEGAL CØMMAND
      IPT = IPP-IP+1
      CALL CLM(0)
      IF(IP.EQ.1) GØTØ 5
      N = 97-IP
      IP = IP-1
      DØ 6 I=1,N
    6 INPUT(I) = INPUT(IP+I)
      DØ 7 I=N+1,96
    7 INPUT(I) = NULL
    5 ENCØDE(96,200,MESSØ) (INPUT(I),I=1,96)
  200 FØRMAT(96A1)
      CALL TEXTØ(IDEV,MESSØ,24,1,1,1,2,IE)
      CALL CLM(0)
      IF(IPT.GE.94) GØTØ 9
      ENCØDE(88,100,MESSØ) (INPUT(I),I=IPT,IPT+3)
  100 FØRMAT('THE SEGMENT BEGINNING WITH *',4A1,'* IS ILLEGAL. CØRRECT',
     1' LINE 1 AS SHØWN AND RESUBMIT IT.  ')
      GØTØ 10
    9 ENCØDE(52,300,MESSØ)
  300 FØRMAT('THE PØINTER HAS PASSED THE END ØF THE INPUT VECTØR  ')
   10 CALL TEXTØ(IDEV,MESSØ,24,2,1,1,2,IE)
      IP = 1
      CALL CLM(0)
      IDIR(1) = IDIR(1)-MØD(IDIR(1),8)
      RETURN
      END
```

```
      SUBROUTINE NUMBER(INIPT,INT,LIMIT,INFORM)
C     INFORM TELLS WHAT TO DO, PACK A NAME OR FIND A NUMBER
C     INIPT IS A POINTER INTO INPUT WHICH TELLS WHERE A NUMBER IS TO
C     APPEAR
C     IF INIPT IS THE NEGATIVE OF THE POINTER A COMMA IS EXPECTED TO
C     FOLLOW THE NUMBER
C     IF INIPT IS POSITIVE A DOLLAR SIGN IS EXPECTED TO FOLLOW THE
C     NUMBER
C     IF THE PROPER CHARACTER IS NOT FOUND INIPT IS SENT BACK AS THE
C     NEGATIVE OF THE POINTER AND INT IS SET TO ZERO
C     IF SOME OTHER KIND OF ERROR IS FOUND INIPT IS SENT BACK AS THE
C     NEGATIVE OF THE POINTER AND INT POINTS TO THE ERROR
C     FOR A NORMAL TERMINATION INT IS SET TO THE NUMBER AND INIPT POINTS
C     TO THE COMMA OR DOLLAR SIGN FOLLOWING THE NUMBER
C     LIMIT IS THE EXPECTED LENGTH OF THE NUMBER IN THE INPUT VECTOR
C
      COMMON/INPUT/INPUT(96),IP,INDIC
      DIMENSION NUMBERS(11)
      DATA NUMBERS,IPLUS,IMINUS,ICOMA,IDOLS/4H0   ,4H1   ,4H2   ,4H3   ,
     1 4H4   ,4H5   ,4H6   ,4H7   ,4H8   ,4H9   ,4H    ,4H+   ,4H-   ,
     2 4H,   ,4H$   /
      INT = 0
      ICHAR = IDOLS
      IF(INIPT.GT.0) GOTO 4
      ICHAR = ICOMA
      INIPT = -INIPT
    4 IF(INFORM.EQ.1) GOTO 16
      DO 5 I=INIPT,INIPT+LIMIT
      IF(INPUT(I).EQ.ICHAR) GOTO 6
    5 CONTINUE
C     FOLLOWING CHARACTER DID NOT APPEAR WITHIN THE LIMIT
      INIPT = -INIPT
      RETURN
    6 NTEN = 1
      DO 7 K=1,I-INIPT
```

```
      KK = I-K
      IF(INPUT(KK).NE.NUMBERS(11)) GOTO 8
    7 CONTINUE
    8 KK = KK+1
      DO 10 J=1,KK-INIPT
      K = KK-J
      DO 11 L=1,11
      IF (INPUT(K).EQ.NUMBERS(L)) GOTO 12
   11 CONTINUE
   12 IF(L.GE.11) GOTO 13
      INT = INT+(L-1)*NTEN
      NTEN = NTEN*10
   10 CONTINUE
      GOTO 15
   13 IF(INPUT(K).NE.NUMBERS(11)) GOTO 14
      K = K-1
      IF(K.LT.INIPT) GOTO 15
      GOTO 13
   14 IF(INPUT(K).EQ.IPLUS) GOTO 15
      IF(INPUT(K).EQ.IMINUS) GOTO 9
C     FALSE CHARACTER IN A NUMBER
      INT = K
      INIPT = -INIPT
      RETURN
    9 INT = -INT
   15 INIPT = I
      RETURN
C     THIS SECTION PACKS THE FIRST FOUR CHARACTERS OF A NAME
   16 DO 17 I=INIPT,96
      IF(INPUT(I).NE.NUMBERS(11)) GOTO 18
   17 CONTINUE
   18 ENCODE(4,19,INT) (INPUT(J),J=I,I+3)
   19 FORMAT(4A1)
      DO 20 J=I,96
      IF(INPUT(J).EQ.ICHAR) GOTO 21
```

```
   20 CØNTINUE
   21 IF(J.LE.I+3) GØTØ 25
C        NØRMAL TERMINATIØN
      INIPT = J
      RETURN
   25 INT = J
C        $ IN THE NAME
      INIPT = -INIPT
      RETURN
      END
```

```
      SUBRØUTINE SCAN
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      CØMMØN/MESSØ/MESSØ(24),IDEV,IDIR(4)
      DIMENSIØN LETTERS(27)
      DATA LETTERS,NULL /4HA   ,4HB   ,4HC   ,4HD   ,4HE   ,4HF   ,
     1 4HG   ,4HH   ,4HI   ,4HJ   ,4HK   ,4HL   ,4HM   ,4HN   ,4HØ   ,
     2 4HP   ,4HQ   ,4HR   ,4HS   ,4HT   ,4HU   ,4HV   ,4HW   ,4HX   ,
     3 4HY   ,4HZ   ,4H    ,77777777B/
C     THIS RØUTINE DETERMINES WHICH INSTRUCTIØN IS TØ BE PRØCESSED NEXT
C     INDIC IS SET TØ 01 FØR THE *ADD* INSTRUCTIØN
C     INDIC IS SET TØ 02 FØR THE *BACK* INSTRUCTIØN
C     INDIC IS SET TØ 03 FØR THE *BØTTØM* INSTRUCTIØN
C     INDIC IS SET TØ 26 FØR THE *CLEAR* INSTRUCTIØN
C     INDIC IS SET TØ 27 FØR THE *DASH* INSTRUCTIØN
C     INDIC IS SET TØ 04 FØR THE *DEFINE* INSTRUCTIØN
C     INDIC IS SET TØ 28 FØR THE *DØNE* INSTRUCTIØN
C     INDIC IS SET TØ 07 FØR THE *ERASE* INSTRUCTIØN
C     INDIC IS SET TØ 05 FØR THE *EXPAND* INSTRUCTIØN
C     INDIC IS SET TØ 06 FØR THE *EXPAND TØ* INSTRUCTIØN
C     INDIC IS SET TØ 08 FØR THE *FRØNT* INSTRUCTIØN
C     INDIC IS SET TØ 09 FØR THE *GET* INSTRUCTIØN
C     INDIC IS SET TØ 10 FØR THE *HIDE* INSTRUCTIØN
C     INDIC IS SET TØ 32 FØR THE *INPUT* INSTRUCTIØN
C     INDIC IS SET TØ 23 FØR THE *INTENSITY* INSTRUCTIØN
C     INDIC IS SET TØ 11 FØR THE *LEFT* INSTRUCTIØN
C     INDIC IS SET TØ 12 FØR THE *LØAD* INSTRUCTIØN
C     INDIC IS SET TØ 13 FØR THE *MØVE* INSTRUCTIØN
C     INDIC IS SET TØ 14 FØR THE *MØVE TØ* INSTRUCTIØN
C     INDIC IS SET TØ 33 FØR THE *ØUTPUT* INSTRUCTIØN
C     INDIC IS SET TØ 15 FØR THE *RIGHT* INSTRUCTIØN
C     INDIC IS SET TØ 16 FØR THE *RØTATE* INSTRUCTIØN
C     INDIC IS SET TØ 17 FØR THE *RØTATE TØ* INSTRUCTIØN
C     INDIC IS SET TØ 20 FØR THE *SHØW* INSTRUCTIØN
C     INDIC IS SET TØ 18 FØR THE *SHRINK* INSTRUCTIØN
C     INDIC IS SET TØ 19 FØR THE *SHRINK TØ* INSTRUCTIØN
```

```
C     INDIC IS SET TØ 29 FØR THE *SØLID* INSTRUCTIØN
C     INDIC IS SET TØ 24 FØR THE *STØP* INSTRUCTIØN
C     INDIC IS SET TØ 21 FØR THE *SUBTRACT* INSTRUCTIØN
C     INDIC IS SET TØ 22 FØR THE *TØP* INSTRUCTIØN
C     INDIC IS SET TØ 30 FØR THE *UNDASH* INSTRUCTIØN
C     INDIC IS SET TØ 34 FØR THE *WRITE* INSTRUCTIØN
C     INDIC IS SET TØ 31 FØR THE *ZAP* INSTRUCTIØN
C     INDIC IS SET TØ 25 WHEN CURRENT INPUT BUFFER HAS BEEN PRØCESSED
C
C     FIND THE FIRST NØNBLANK CHARACTER
  100 IF(INPUT(IP).NE.LETTERS(27)) GØTØ 200
      IP = IP+1
      IF(IP.EQ.97) GØTØ 80
      GØTØ 100
C     DETERMINE FIRST LETTER ØF INSTRUCTIØN
  200 DØ 30 INDIC=1,27
      IF(INPUT(IP).EQ.LETTERS(INDIC)) GØTØ 40
   30 CØNTINUE
C     ILLEGAL INSTRUCTIØN
   27 CALL ERR(IP)
   60 IF(MØD(IDIR(1),8)) 50,60,50
   50 CALL CLM(0)
      CALL TEXTØ(IDEV,MESSØ,24,2,1,1,2,IE)
      CALL CLM(1)
      IP = 1
      GØTØ 100
   40 GØTØ (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
     1 24,25,26,27),INDIC
    1 INDIC = 1
      GØTØ 90
    2 IF(INPUT(IP+1).EQ.LETTERS(15)) GØTØ 210
      INDIC = 2
      GØTØ 90
  210 INDIC = 3
      GØTØ 90
```

```
  3 INDIC = 26
    GØTØ 90
  4 INDIC = 27
    IF(INPUT(IP+1).EQ.LETTERS(5)) INDIC = 4
    IF(INPUT(IP+1).EQ.LETTERS(15)) INDIC = 28
    GØTØ 90
  5 IF(INPUT(IP+1).NE.LETTERS(24)) GØTØ 51
    INDIC = 1
    GØTØ 70
 52 INDIC = 5+IE
    GØTØ 90
 51 IF(INPUT(IP+1).NE.LETTERS(18)) GØTØ 27
    IF(INPUT(IP+2).NE.LETTERS(1)) GØTØ 27
    INDIC = 7
    GØTØ 90
  6 INDIC = 8
    GØTØ 90
  7 INDIC = 9
    GØTØ 90
  8 INDIC = 10
    GØTØ 90
  9 IF(INPUT(IP+1).NE.LETTERS(14)) GØTØ 27
    INDIC = 23
    IF(INPUT(IP+2).EQ.LETTERS(16)) INDIC=32
    GØTØ 90
 10 GØTØ 27
 11 GØTØ 27
 12 IF(INPUT(IP+1).EQ.LETTERS(15)) GØTØ 121
    INDIC = 11
    GØTØ 90
121 INDIC = 12
    GØTØ 90
 13 INDIC = 2
    GØTØ 70
131 INDIC = 13+IE
```

```
      GØTØ 90
   14 GØTØ 27
   15 INDIC = 33
      GØTØ 90
   16 GØTØ 27
   17 GØTØ 27
   18 IF(INPUT(IP+1).EQ.LETTERS(15)) GØTØ 181
      INDIC = 15
      GØTØ 90
  181 INDIC = 3
      GØTØ 70
  182 INDIC = 16+IE
      GØTØ 90
   19 IF(INPUT(IP+1).EQ.LETTERS(21)) GØTØ 193
      IF(INPUT(IP+1).EQ.LETTERS(8)) GØTØ 192
      IF(INPUT(IP+1).EQ.LETTERS(20)) GØTØ 194
      IF(INPUT(IP+1).EQ.LETTERS(15)) GØTØ 195
      GØTØ 27
  196 INDIC = 4
      GØTØ 70
  191 INDIC = 18+IE
      GØTØ 90
  192 IF(INPUT(IP+2).EQ.LETTERS(18)) GØTØ 196
      INDIC = 20
      GØTØ 90
  193 INDIC = 21
      GØTØ 90
  194 IF(INPUT(IP+2).NE.LETTERS(15)) GØTØ 27
      IF(INPUT(IP+3).NE.LETTERS(16)) GØTØ 27
      INDIC = 24
      GØTØ 90
  195 INDIC = 29
      GØTØ 90
   20 INDIC = 22
      GØTØ 90
```

```
   21 INDIC = 30
      G0T0 90
   22 G0T0 27
   23 INDIC = 34
      G0T0 90
   24 G0T0 27
   25 G0T0 27
   26 INDIC = 31
      G0T0 90
C     CHECK T0 SEE IF THE INSTRUCTI0N IS A *T0* INSTRUCTI0N
   70 I = IP
      CALL BLANKS(I,0)
      CALL BLANKS(I,1)
      IE = 0
      IF(INPUT(I).EQ.LETTERS(20)) IE=1
      G0T0 (52,131,182,191),INDIC
   80 IDIR(1) = IDIR(1)-M0D(IDIR(1),8)
      INDIC = 25
   90 RETURN
      END
```

```
      SUBRØUTINE SETVEW(IVIEW,KEY,IGNØRE,I)
C     THIS SUBRØUTINE SETS BITS TØ INDICATE WHICH VIEWS A LINE WILL
C     SHØW WITH
C     BIT 0 *SIDE ALØNE* 2**0 = 1
C     BIT 1 *WITH FRØNT* 2**1 = 2
C     BIT 2 *WITH FRØNT AND LEFT* 2**2 = 4
C     BIT 3 *WITH FRØNT AND RIGHT* 2**3 = 8
C     BIT 4 *WITH FRØNT AND TØP* 2**4 = 16
C     BIT 5 *WITH FRØNT AND BØTTØM* 2**5 = 32
C     BIT 6 *WITH BACK* 2**6 = 64
C     BIT 7 *WITH BACK AND LEFT* 2**7 = 128
C     BIT 8 *WITH BACK AND RIGHT* 2**8 = 256
C     BIT 9 *WITH BACK AND TØP* 2**9 = 512
C     BIT 10 *WITH BACK AND BØTTØM* 2**10 = 1024
C     BIT 11 *WITH LEFT* 2**11 = 2048
C     BIT 12 *WITH LEFT AND TØP* 2**12 = 4096
C     BIT 13 *WITH LEFT AND BØTTØM* 2**13 = 8192
C     BIT 14 *WITH RIGHT* 2**14 = 16384
C     BIT 15 *WITH RIGHT AND TØP* 2**15 = 32768
C     BIT 16 *WITH RIGHT AND BØTTØM* 2**16 = 65536
C     BIT 17 *WITH TØP* 2**17 = 131072
C     BIT 18 *WITH BØTTØM* 2**18 = 262144
C
C     IVIEW(1) TELLS WHICH VIEWS FRØNT ØR BACK
C     IVIEW(2) TELLS WHICH VIEWS RIGHT ØR LEFT
C     IVIEW(3) TELLS WHICH VIEWS TØP ØR BØTTØM
C     KEY IS THE NUMBER TØ SET THE BITS IN
C     IGNØRE TELLS WHICH WHICH IVIEW IS TØ BE LEFTØUT
C     I TELLS WHETHER TØ ADD ØR SUBTRACT TØ MAKE A BIT 1 ØR 0
C
      DIMENSIØN IVIEW(3)
      GØTØ (100,200,300),IGNØRE
C     WE ARE ØN A FRØNT ØR BACK VIEW
  100 IF(IVIEW(2)-1) 110,130,120
  110 IF(IVIEW(3)-1) 111,112,113
```

```
  111 KEY = KEY+1*I
      GØTØ 400
  112 KEY = KEY+131072*I
      GØTØ 400
  113 KEY = KEY+262144*I
      GØTØ 400
  120 IF(IVIEW(3)-1) 121,122,123
  121 KEY = KEY+2048*I
      GØTØ 400
  122 KEY = KEY+4096*I
      GØTØ 400
  123 KEY = KEY+8192*I
      GØTØ 400
  130 IF(IVIEW(3)-1) 131,132,133
  131 KEY = KEY+16384*I
      GØTØ 400
  132 KEY = KEY+32768*I
      GØTØ 400
  133 KEY = KEY+65536*I
      GØTØ 400
C     WE ARE ØN THE LEFT ØR RIGHT SIDE VIEW
  200 IF(IVIEW(1)-1) 210,220,230
  210 IF(IVIEW(3)-1) 111,112,113
  220 IF(IVIEW(3)-1) 221,222,223
  221 KEY = KEY+2*I
      GØTØ 400
  222 KEY = KEY+16*I
      GØTØ 400
  223 KEY = KEY+32*I
      GØTØ 400
  230 IF(IVIEW(3)-1) 231,232,233
  231 KEY = KEY+64*I
      GØTØ 400
  232 KEY = KEY+512*I
      GØTØ 400
```

```
  233 KEY = KEY+1024*I
      G0T0 400
C     WE ARE 0N A T0P 0R B0TT0M VIEW
  300 IF(IVIEW(1)-1) 310,320,330
  310 IF(IVIEW(2)-1) 111,131,121
  320 IF(IVIEW(2)-1) 221,323,322
  322 KEY = KEY+4*I
      G0T0 400
  323 KEY = KEY+8*I
      G0T0 400
  330 IF(IVIEW(2)-1) 231,333,332
  332 KEY = KEY+128*I
      G0T0 400
  333 KEY = KEY+256*I
  400 RETURN
      END
```

```
      SUBRØUTINE SHØW(IPP,INT)
C     THIS SUBRØUTINE SETS BITS INT THE INTEGER INT ACCØRDING TØ A SHØW-
C     HIDE GRØUP FØR A LINE
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      DIMENSIØN LETTERS(10),IVIEW(3)
      DATA LETTERS,IHIDE/4HS    ,4HH    ,4HF    ,4HB    ,4HR    ,4HL    ,
     14HT    ,4HA    ,4HI    ,4HØ    ,07777777B/
C     SHØW DETERMINES IF THERE IS A HIDE ØR SHØW INSTRUCTIØN
C     IPP IS SENT BACK AS -IPP IF THERE IS AN ERRØR
C     ASSUME THERE IS A SHØW GRØUP
      MARK = 1
      INT = 0
      INFØRM = 2
C     INFØRM IS USED AS AN INDICATØR FØR THE BLANKS RØUTINE
      CALL BLANKS(IPP,1)
      IF(INPUT(IPP).NE.LETTERS(2)) GØTØ 500
C     HIDE GRØUP
      MARK = -1
      INT = IHIDE
      GØTØ 497
  500 IF(INPUT(IPP).EQ.LETTERS(1)) GØTØ 497
C     NØT A HIDE ØR SHØW GRØUP
      IPP = -IPP
      RETURN
  497 CALL BLANKS(IPP,0)
  498 DØ 499 I=1,3
      IVIEW(I) = 0
  499 CØNTINUE
  501 CALL BLANKS(IPP,1)
      IF(INPUT(IPP).EQ.LETTERS(9)) GØTØ 510
      DØ 502 I=1,6
      IF(INPUT(IPP).EQ.LETTERS(I+2)) GØTØ 503
  502 CØNTINUE
C     ILLEGAL VIEW
      IPP = -IPP
```

```
      RETURN
  503 IF(I.NE.6) GØTØ 504
      INT = 0
      IF(MARK.EQ.1) INT = IHIDE
      CALL BLANKS(IPP,INFØRM)
      IPP = -IPP
      RETURN
  504 IF(I.NE.2) GØTØ 509
      IF(INPUT(IPP+1).EQ.LETTERS(10)) I=6
  509 DØ 505 J=2,6,2
      IF(I.LE.J) GØTØ 506
  505 CØNTINUE
  506 IVIEW(J/2) = 1
      IF(I.EQ.J) IVIEW(J/2)=2
  510 CALL BLANKS(IPP,INFØRM)
      IF(IPP.LT.0) GØTØ 507
      GØTØ 501
  507 CALL SETVEW(IVIEW,INT,INDIC,MARK)
      IF(INFØRM.LT.0) GØTØ 508
      IPP = -IPP+1
      GØTØ 498
  508 IPP = -IPP
      RETURN
      END
```

```
      SUBRØUTINE STØALØ(INITIAL,LENGTH,INFØRM)
C     INITIAL TELLS FIRST PØINT IN THE LIST TØ BE ØPERATED ØN
C     LENGTH TELLS NUMBER ØF PØINTS TØ BE ØPERATED ØN
C     INFØRM TELLS WHAT TØ DØ
C     INFØRM = 1   MØVE STØRAGE
C     INFØRM = 2   ERASE STØRAGE
C     INFØRM = 3   ADD STØRAGE
      CØMMØN/IFIGUR/IFIGUR(5000),IFP,IFIG
      IBEG = INITIAL
      ILEN = LENGTH
      GØTØ (100,200,300),INFØRM
  100 NEXT = IFIGUR(IFIG+1)
      IF(IFP+ILEN.LE.5000) GØTØ 101
C     THERE IS NØT ENØUGH STØRAGE INFØRM IS SET TØ 0
      INFØRM = 0
      RETURN
  101 IFIGUR(IFIG+1) = 0
      IF(NEXT.EQ.0) GØTØ 400
  105 DØ 106 I=3,8
      IFIGUR(NEXT+I) = IFIGUR(NEXT+I)-ILEN
  106 CØNTINUE
      IF(IFIGUR(NEXT+1).EQ.0) GØTØ 110
      IFIGUR(NEXT+1) = IFIGUR(NEXT+1)-IFIGUR(IFIG+2)
      NEXT = IFIGUR(NEXT+1)+IFIGUR(IFIG+2)
      GØTØ 105
  110 IFIGUR(NEXT+1) = IFP-IFIGUR(IFIG+2)
      IFIGUR(IFIG+1) = 0
      IFP = IFP-1
      IFIG = IFIG-1
      DØ 120 I=1,ILEN
      IFIGUR(IFP+I) = IFIGUR(IFIG+I)
  120 CØNTINUE
      IFIG = IFIGUR(NEXT+1)-IFIG-1
      DØ 122 I=4,9
      IFIGUR(IFP+I) = IFIGUR(IFP+I)+IFIG
```

```
122 CØNTINUE
    IFIG = IFIGUR(NEXT+1)
    IFP = IFP+1+ILEN
200 IFP = IFP-ILEN
    DØ 210 I=IBEG,IFP-1
    IFIGUR(I) = IFIGUR(I+ILEN)
210 CØNTINUE
    RETURN
300 IF(IFP+ILEN.LE.5000) GØTØ 310
    INFØRM = 0
    GØTØ 400
310 DØ 320 I=1,IFP-IBEG
    IFIGUR(IFP+ILEN-I) = IFIGUR(IFP-I)
320 CØNTINUE
    IFP = IFP+ILEN
400 RETURN
    END
```

```
      SUBRØUTINE VIEW(IPP)
C     THIS SUBRØUTINE TELLS WHICH VIEW IS BEING CØNSIDERED IN AN INST
C     INDIC IS SET TØ 1 FØR FRØNT
C     INDIC IS SET TØ 2 FØR BACK
C     INDIC IS SET TØ 3 FØR RIGHT
C     INDIC IS SET TØ 4 FØR LEFT
C     INDIC IS SET TØ 5 FØR TØP
C     INDIC IS SET TØ 6 FØR BØTTØM
C     INDIC IS SET TØ 0 IF THERE IS AN ERRØR
      CØMMØN/INPUT/INPUT(96),IP,INDIC
      DIMENSIØN LETTERS(6)
      DATA LETTERS/4HF   ,4HB   ,4HR   ,4HL   ,4HT   ,4HØ   /
      DØ 10 I=1,5
      IF(INPUT(IPP).EQ.LETTERS(I)) GØTØ 20
   10 CØNTINUE
C     ERRØR FØUND
      INDIC = 0
      RETURN
   20 INDIC = I
      IF(INDIC.NE.2) RETURN
      IF(INPUT(IPP+1).EQ.LETTERS(6)) INDIC = 6
      RETURN
      END
```

```
      SUBROUTINE WRITE(IFIG1)
C     THIS SUBROUTINE IS USED TO LIST THE INTERNAL STRUCTURE OF A FIGURE
C     ON THE LINE PRINTER
      COMMON/IFIGUR/IFIGUR(5000),IFP,IFIG
      DIMENSION ISHOW(19),IFACE(12)
      DATA IFACE,IBLANK,I1/4HFRON,4HT   ,4HBACK,4H    ,4HRIGH,4HT   ,
     1 4HLEFT,4H    ,4HTOP ,4H    ,4HBOTT,4HOM  ,4H    ,4H1   /
      WRITE(6,100) IFIGUR(IFIG1),IFIGUR(IFIG1+9)
  100 FORMAT(1H1,64X,A4,//,58X,'SCALE =',I10,3/)
      DO 50 II=1,6
      WRITE(6,200) IFACE(II+II-1),IFACE(II+II)
  200 FORMAT(' FACE-',2A4,5X,'A 1 IS USED TO INDICATE WHEN A LINE WILL'
     1 ,' BE SHOWN',//,6X,'X',9X,'Y',9X,'Z',9X,' BOT.',' TOP.','R BO.',
     2 ' R T.',' RIG.','L BO.',' L T.',' LEF.','BA BO','.BA T','.BA R',
     3 '.BA L','.BAC .',' F BO','.F T.',' F R.',' F L.',' FRO.',' ITS')
      J = IFIGUR(IFIG1+2+II)+1
      JJ = IFIGUR(IFIG1+3+II)
      IF(II.EQ.6) JJ=IFIG1+IFIGUR(IFIG1+2)
   10 N2 = 1
      DO 20 K=1,19
      L = J+3
      ICK = IFIGUR(L)/N2
      JCK = ICK/2
      JCK = JCK+JCK
      ISHOW(20-K) = IBLANK
      IF(JCK.NE.ICK) ISHOW(20-K)=I1
      N2 = N2*2
   20 CONTINUE
      WRITE(6,300) (IFIGUR(K),K=J,J+2),(ISHOW(K),K=1,19)
  300 FORMAT(/,1X,2(I9,','),I9,6X,19(2X,A1,2X))
      J = J+4
      IF(J.LT.JJ) GOTO 10
      IF(II.EQ.6) GOTO 50
      WRITE(6,400)
  400 FORMAT(1H1)
```

```
50 CONTINUE
   RETURN
   END
```

REFERENCES

1. Boehm, V. R., and others, POGO: Programmer-Oriented Graphics Operation, The Rand Corporation, 1969.

2. Brown, G. D., and Bush, C. H., The Integrated Graphics System for the IBM 2250, The Rand Corporation, 1968.

3. Ellis, T. O., and Sibley, W. L., On the Development of Equitable Graphics I/O, The Rand Corporation, 1966.

4. Feder, J., Linguistic Specification and Analysis of Classes of Line Patterns, Ph.D. Thesis, New York University, Bronx, New York, 1969.

5. Hodes, L., "A Programming System for the On-Line Analysis of Biomedical Images," Communications of the ACM, v. 13, pp. 279-283, May 1970.

6. Hodgman, C. D., editor, C. R. C. Standard Mathematical Tables, pp. 403-417, Chemical Rubber Publishing Company, 1961.

7. International Business Machines Corporation, IBM SYSTEM/360 Operating System Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL, and PL/I, 1969

8. Scientific Data Systems, FORTRAN IV REFERENCE MANUAL, 1966. Note: Scientific Data Systems has since been renamed Xerox Data Systems.

9. Siders, R. A., and others, Computer Graphics, American Management Association, 1966.

10. Thornhill, D. E., and others, An Integrated Hardware-Software System for Computer Graphics in Time-Sharing, Massachusetts Institute of Technology, 1968.

11. Xerox Data Systems, Naval Postgraduate School Display Subsystem.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Asst Professor G. L. Barksdale, Code 53 Bv Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
4.	LTJG Scott H. Mayer, USN 6N260 Rosedale Road Roselle, Illinois 60172	1

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

INATING ACTIVITY (Corporate author) val Postgraduate School nterey, California 93940	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

ORT TITLE

COPE: AN INTRODUCTORY GRAPHICS LANGUAGE

CRIPTIVE NOTES (Type of report and, inclusive dates)

aster's Thesis; June 1970

HOR(S) (First name, middle initial, last name)

Scott Hilmar Mayer
ieutenant, junior grade, United States Navy

ORT DATE une 1970	7a. TOTAL NO. OF PAGES 98	7b. NO. OF REFS 11
NTRACT OR GRANT NO. OJECT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)	
	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

TRIBUTION STATEMENT

is document has been approved for public release and sale; its stribution is unlimited.

PPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940

STRACT

The SCOPE language has been designed to provide an introduction interactive computing and the cathode-ray tube graphics display. e user is given the opportunity to input a figure and see the nds of things that can be done with that figure on the display reen. The language has been implemented on an XDS 9300 computer terfaced with an Adage AGT 10 graphics terminal. Each instruction described, and the algorithms used to actually display figures e also described. Suggestions for future implementations are also cluded.

lassified
rity Classification

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
puter Language						
puter Graphics						
phics						
eractive Computing						

..1473 (BACK)
·-6321

Unclassified
Security Classification

A-31409